(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

TACTICAL DECISION MAKING IN INTELLIGENT AGENTS: DEVELOPING AUTONOMOUS FORCES IN NPSNET

by

Michael E. Culpepper

March 1992

Thesis Advisor: Hemant K. Bhargava

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (*If applicable*) AS	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (*If applicable*)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (*City, State, and ZIP Code*)	10 SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11. TITLE (*Include Security Classification*)
TACTICAL DECISION MAKING IN INTELLIGENT AGENTS.
DEVELOPING AUTONOMOUS FORCES IN NPSNET

12. PERSONAL AUTHOR(S) Culpepper, Michael E.

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (*year, month, day*)	15 PAGE COUNT
Master's Thesis	From To	March 1992	87

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Autonomous Agents, Autonomous Forces, Artificial Intelligence, Rule Based Systems, Combat Modeling, Combat Simulations, Tactical Decision Making Models

19 ABSTRACT (*continue on reverse if necessary and identify by block number*)

This thesis presents a conceptual framework, system architecture, and working prototype for a tactical decision making model. This model was developed within the context of intelligent autonomous forces in combat modeling systems. The goal of this model is to realistically portray the behavior of tactical units operating on the battlefield.

In our prototype, tactical decision making principles and heuristics are modeled as rules in a logic programming system, and are implemented in an expert system development environment. The current implementation plans, executes, and monitors its decisions in real-time during the course of the combat simulation. This research also examines several challenges in the modeling and execution of tactical-level decisions of an autonomous force. This thesis is a significant first step in developing fully automated forces that can do human tactical decision making.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS REPORT ☐ DTIC USERS	UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Hemant K. Bhargava	22b TELEPHONE (*Include Area code*) (408) 646 2264	22c OFFICE SYMBOL AS

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited.

Tactical Decision Making in Intelligent Agents:
Developing Autonomous Forces in NPSNET

by

Michael E. Culpepper
Captain, United States Army
B.S., Henderson State University, 1982

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
MARCH 1992

Author: _____
Michael E. Culpepper

Approved by: _____
Hemant K. Bhargava, Thesis Advisor

_____
David R. Pratt, Second Reader

_____
David R. Whipple, Jr., Chairman
Department of Administrative Sciences

ABSTRACT

This thesis presents a conceptual framework, system architecture, and working prototype for a tactical decision making model. This model was developed within the context of intelligent autonomous forces in combat modeling systems. The goal of this model is to realistically portray the behavior of tactical units operating on the battlefield.

In our prototype, tactical decision making principles and heuristics are modeled as rules in a logic programming system, and are implemented in an expert system development environment. The current implementation plans, executes, and monitors its decisions in real-time during the course of the combat simulation. This research also examines several challenges in the modeling and execution of tactical-level decisions of an autonomous force. This thesis is a significant first step in developing fully automated forces that model human tactical decision making.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. COMBAT SIMULATION SYSTEMS 2

B. MOTIVATION FOR AUTONOMOUS FORCES IN COMBAT SIMULATORS 3

C. INTELLIGENT AGENTS: OBSERVATION AND DECISION MAKING 4

1. Observation 5

2. Decision Making and Execution 7

D. SCOPE OF THESIS 8

II. TACTICAL DECISION MAKING IN A COMBAT ENVIRONMENT . 10

A. TACTICAL DECISION MAKING: LEVELS OF RESPONSIBILITY 10

1. Individual Level 11

2. Crew Level 11

3. Unit Level 12

B. FUNCTIONAL AREAS IN COMBAT 12

1. Command and Control 12

2. Movement and Route Planning 13

3. Target Engagement 13

C. BATTLEFIELD DECISION MAKING TASKS 14

1. Terrain Assessment 14

2. Target Assessment 15

3. Target Acquisition 15
4. Fire Control 15
5. Concept of Operations Development 16
6. Coordination and Communication 16
D. NPSNET AF DECISION MAKING MODULE ARCHITECTURE . 16
III. AF COMMAND AND CONTROL 18
A. TACTICAL CONCEPT DEVELOPMENT 19
1. Mission Guidance 20
2. Enemy Forces 20
3. Friendly Forces 21
4. Terrain 21
B. ASSIGNMENT OF TASKS TO SUBORDINATE UNITS . . . 22
C. TARGET ASSESSMENT AND ASSIGNMENT 24
1. Platoon Target Assignment 25
2. Individual Tank Target Assignment 26
D. COMMUNICATION 27
IV. AF MOVEMENT AND ROUTE PLANNING 29
A. ROUTE PLANNING 29
B. PLATOON MOVEMENT 31
C. INDIVIDUAL VEHICLE MOVEMENT 32
V. TARGET ENGAGEMENT 35
A. SECTOR SCANNING 35
B. TARGET ACQUISITION 35
C. FIRE CONTROL 36
1. Main Gun Alignment 37
2. Hit Probability Assessment 37

3. Line of Sight Check 38
VI. IMPLEMENTATION 40
A. PROGRAM DESCRIPTION 40
1. AF Initialization Module 41
2. Belief Generation Module 41
3. Tactical Decision Making Module 42
B. EXECUTION CYCLE 42
1. Read Update Messages from Network 42
2. Convert World Information to Beliefs 43
3. Update Data Structures 43
4. Generate Tactical Decisions 44
5. Send AF Orders to Simulator 44
C. IMPLEMENTATION ASSESSMENT 44
VII. CONCLUSIONS AND RECOMMENDATIONS 47
A. CONTRIBUTIONS OF THIS RESEARCH 47
B. ASSESSMENT OF CURRENT AF MODEL 48
C. RECOMMENDED FOLLOW ON WORK 49
APPENDIX A - NPSNET/AF REPRESENTATION CONVENTIONS . . . 52
APPENDIX B - FIRE DISTRIBUTION CALCULATIONS 54
APPENDIX C - MOVEMENT CALCULATIONS 56
APPENDIX D - TARGET ENGAGEMENT CALCULATIONS 61
APPENDIX E - AF DECISION MAKING MODULE SOURCE CODE . . 66
LIST OF REFERENCES 77
INITIAL DISTRIBUTION LIST 79

I. INTRODUCTION

This thesis presents a conceptual framework, system architecture, and working prototype for a tactical decision making model. This model was developed within the context of intelligent autonomous forces in combat modeling systems. The goal of this model is to realistically portray the behavior of tactical units operating on the battlefield. An autonomous force in a combat modeling system employs battlefield information, tactical judgment, and mission requirements to make decisions directed toward the satisfaction of its overall objectives. The force plans, executes, and monitors its decisions in real-time during the course of the simulation. The behavior of such autonomous forces is tested against forces controlled by human players in the combat modeling system.

Tactical decision making principles and heuristics are modeled as rules in a logic programming system, and are implemented in an expert system development environment. The autonomous force uses its tactical decisions to develop an executable, operational plan that directs its actions on the battlefield. This research also examines several challenges in the modeling and execution of tactical-level decisions of an autonomous force. This thesis is a significant first step in developing fully automated, intelligent forces that model

tactical decision making. The rest of this chapter introduces background material and discusses the motivation and objectives for this research.

A. COMBAT SIMULATION SYSTEMS

Combat simulation systems provide today's military leaders with one of the most cost effective training tools available. These devices allow military personnel to practice their art without having to incur the costs associated with the operation of their combat equipment. Additionally, training simulators allow soldiers to practice tasks that are inherently hazardous without the risks associated with live-fire or maneuver training. Combat simulators are also used for the development, evaluation, and validation of new weapons systems, tactics, and doctrine. As defense dollars become scarcer it becomes increasingly important to develop realistic combat simulators. [Ref. 1]

The Computer Science Department at the Naval Postgraduate School is currently developing the NPSNET combat simulator. NPSNET is a real-time, interactive, visual combat simulation system. The goal of this system is to create a virtual world for combat training, planning, and gaming [Ref. 2]. NPSNET displays terrain and man-made features, as well as moving aircraft, ships, and ground vehicles in 3D computer graphics. Each user of the system operates a vehicle from one of the networked workstations and is presented with a graphical

representation of the world from that vehicle's perspective. Vehicle control commands from each workstation are transmitted over an Ethernet to all other workstations on the network. This allows multiple users to interact with one another. In addition to the user operated vehicles, NPSNET supports unmanned vehicles executing scripted actions and autonomous forces which interact with the users.

An important component of a combat simulation system is the "simulator." The simulator acts as a monitor and referee, and consists of programs that determine the state of any object in the system at any time. Thus. it determines the nature of the battlefield terrain, exact locations of various fcrces, amounts and types of munitions they have, whether any forces are destroyed as a result of an engagement, and so on. The simulator presents much of this information using 3D graphics to the human players participating in the simulation. Thus, the simulator must possess true information about all relevant attributes of all the static and moving objects in the system.

B. MOTIVATION FOR AUTONOMOUS FORCES IN COMBAT SIMULATORS

The availability of autonomous forces provides an extra dimension of flexibility and functionality in combat simulation systems. The use of autonomous forces allows users to operate the system with reduced support personnel requirements. A combat simulator is used to support specific

user determined training objectives. To achieve these training objectives, the user will usually create a scenario including other friendly units operating in the area as well as opposing forces. In the absence of autonomous forces, adjacent friendly units and opposing forces must be operated by other human players in support of the user's training. The use of autonomous forces reduces the personnel requirements for a training session and hence the cost. Additionally, autonomous forces operate in a controllable and consistent manner to ensure that a user's desired scenario is achieved. The degree to which the user is challenged by autonomous opposing forces is not dependent upon the skill of other support players, but is instead prescribed by the operating characteristics of the autonomous force.

C. INTELLIGENT AGENTS: OBSERVATION AND DECISION MAKING

This research addresses the development of an autonomous force as an intelligent agent within the NPSNET system. Central to the development of this autonomous force is the idea that the behavior of a combat force can be modeled by considering two broad categories of functions--observation and decision making--performed by such a force. This is similar to the separation of intelligent agent functions into the categories of perception and action [Ref. 3]. The combat force can be thought of as consisting of an "observer" (whose role is to gather relevant information about the combat

environment) and a "decision maker" (whose role is to make suitable decisions by combining this information with its tactical judgment and mission requirements). This, then, leads to a computational model of an autonomous force wherein the observer and decision maker are modeled with separate, independent programs. The role of these two programs is briefly described below, but the focus of this thesis is on the latter.

1. Observation

Intelligent agents acquire information about their world through sensors of one type or another. An agent that operates in the real world uses its "eyes and ears," radar, sonar, infrared, video, and other sensors to gather information about its physical surroundings. The information gathered is only as accurate as the sensors are in the given environmental conditions. Thus, we can make a distinction between true *knowledge* about the world and the *beliefs* that an intelligent agent forms about its world. The accuracy of these beliefs is constrained by the observation capabilities of the agent. For example, an agent might be unable to measure accurately the speed of a target five kilometers away, and may have no information at all regarding the target's ammunition status. Yet, the truth about these attributes must be known to the simulator for it to be able to act as a competent monitor and referee.

At a simplistic level, an autonomous force in a combat simulation system does not need external sensors; it could simply receive information about the terrain and enemy forces from the simulator. However, such a situation is not desirable for the following reasons. First, it would amount to the autonomous force being supported by a biased simulator, since it would always know the truth about all relevant attributes. It would give the autonomous force a distinct advantage over human players in the simulator. Second, it would fail to provide a realistic model of a tactical force since it would not model the observation abilities of this force.

The solution to this problem in our research is to separate the observation functions of an autonomous force from its decision making functions and to develop an observation model. The observer module of this force is then tasked with monitoring the world and introducing a degree of error to world knowledge. The output of this module is a set of observations that constitutes the autonomous force's beliefs about the world. The details of this module and the principles underlying the transformation of world knowledge to belief are discussed in [Ref. 4,5]. The objective of this model is to provide the autonomous force with observations that are consistent with the capabilities of its simulated personnel and equipment and the current environmental conditions.

2. Decision Making and Execution

The decision making function, which is the subject of this thesis, determines what actions to take based on the information made available to it. If this information consists of beliefs provided by an observer module, the decisions must be made on the basis of these beliefs, perhaps accounting for the incompleteness and inaccuracy of the information. In the absence of an error-introducing observer, decisions are made with full knowledge of the true world state. In either case, an intelligent agent must respond continuously, in a manner consistent with its objectives, to information about its environment. It must use its judgment and information about its objectives to make decisions and develop or revise goals. The intelligent agent must then transform these decisions into an executable plan that specifies its actions in response to newly acquired information. Finally, the intelligent agent must simulate the execution of these responses in the combat simulation system.

Various decision making approaches for intelligent agents are presented in [Ref. 6]. The approaches generally fall into two categories, scripted and reactive [Ref. 7]. Scripted plans involve the prescription of a set sequence of actions to follow in order to achieve a goal. This approach is best suited to agents that operate in a static environment. Intelligent agents that operate in a dynamic environment must use a reactive approach. In this approach the agent

improvises as its environment changes. This improvisation involves the meta-level selection of plans that are applicable to the situation. For an autonomous force in a combat simulator a reactive approach is required. This is the approach presented in this thesis.

D. SCOPE OF THESIS

The goal of this thesis is to develop a viable framework for the Autonomous Force (AF) in the NPSNET simulator. This thesis focuses on the decision making and implementation functions described in Section C. A companion thesis [Ref. 5] addresses the observation function. The framework developed was kept as general as possible to facilitate its use with a variety of missions and unit types. The objectives for this thesis are listed below.

- Develop a model for an autonomous force that operates in a manner consistent with its weapons system capabilities and current tactical principles
- Model the multi-echelon nature of tactical decision making
- Model procedures that coordinate the actions of multiple agents toward an overall common goal
- Integrate observer and decision making models into a single autonomous force model
- Implement a working prototype autonomous force model in the NPSNET simulator

The specific prototype model that was implemented is for a tank company conducting offensive or reconnaissance operations against the human players in the simulator. Where specific

operating characteristics were required, M1A1 Tank specifications were used [Ref. 8].

II. TACTICAL DECISION MAKING IN A COMBAT ENVIRONMENT

This chapter describes the aspects of tactical decision making that must be addressed to model this process in a computer program. The first step in developing the decision making module was to examine how tactical decision making actually occurs. This involves consideration of both the division of responsibilities among various echelons within a unit and an examination of the functional areas that contribute to tactical decision making. While the decision making module does not attempt to mirror the human decision making process, it does attempt to produce comparable results. The study of actual decision making procedures provides a rough outline for the structure of the program. [Ref. 9] and [Ref. 10] provide detailed presentations of the military decision making process. This chapter discusses those aspects of tactical decision making that are modeled in the AF program.

A. TACTICAL DECISION MAKING: LEVELS OF RESPONSIBILITY

Battlefield decision making can be categorized into three basic levels: individual, crew, and unit. On the battlefield these are very distinct levels of responsibility. In attempting to model this process in a computer program the distinction becomes a little less pronounced in some cases but

in general still provides a good framework. This section discusses each of these levels.

1. Individual Level

The individual level refers to those decisions that an individual soldier makes on a minute by minute basis. These decisions guide a soldier's actions as he manipulates an assigned weapon or piece of equipment. In general, these actions are fairly procedural in nature and involve the analysis of only a limited amount of information and alternatives.

2. Crew Level

The individual soldiers that are assigned to the various positions required to operate a weapons system are collectively referred to as a crew. The crew commander coordinates the efforts of the soldiers assigned to his crew in order to accomplish the missions that have been assigned to the crew. Concurrent individual actions, such as the steering of a tank and the rotation of the tank's turret, are coordinated to achieve the desired overall effect. The crew commander has many more decision factors to analyze and a much broader array of alternatives to consider than is found at the individual level. This is also the first level at which actions are required to coordinate the efforts of multiple agents. At the crew level the decision making process becomes more complex.

3. Unit Level

The unit level refers to those actions performed by a unit's leadership to coordinate and control the actions of all assets assigned to the unit. Unit level actions occur at each echelon above the crew level. This is the most complex of the three levels and involves the analysis of large amounts of situational information and the consideration of virtually limitless alternatives. Additionally, whereas the lower levels deal primarily with deciding *how* to do things, the unit level must also decide *what* things to do.

B. FUNCTIONAL AREAS IN COMBAT

The functional areas to be addressed increase significantly in number and complexity as decision making is modeled at higher echelons of command. The focus of this thesis is at the lower echelons (company and below) and will be limited to the subset of areas necessary for the AF to conduct operations within NPSNET. Specifically this thesis will address tactical command and control, movement and route planning, and target engagement. This section introduces these three areas which are discussed in detail in Chapters III, IV, and V.

1. Command and Control

Command and control functions generally correspond to actions performed at the unit level. Command and control decisions address the following:

- Development of the tactical concept of operations
- Assignment of tasks to subordinate units
- Target assessment and assignment
- Communication of information
- Coordination and synchronization of assets

2. Movement and Route Planning

This functional area addresses all aspects of how to move from a given location to an assigned movement objective. The decisions made here correspond to those that leaders and vehicle drivers make in deciding how to execute a movement order. Given an assigned march objective, the element leader must assess his current location, the location of his objective, and the intervening terrain in order to select an appropriate route. Given an assigned route, the vehicle driver must make the continuous steering decisions necessary to follow the route.

3. Target Engagement

Target engagement covers all actions necessary to bring a unit's weapons to bear on assigned targets. Decisions made at all three levels provide input to this process. Target engagement addresses the following:

- Target analysis
- Target acquisition
- Distribution of fires among multiple targets
- Firing decision

C. BATTLEFIELD DECISION MAKING TASKS

This section discusses some of the specific battlefield tasks that must be addressed in order to model the decision making process. These tasks do not correspond to one specific decision making level. For some tasks, various aspects of the task are performed concurrently at different levels. In other cases, the same task may be performed at multiple levels but for different purposes dictated by the needs of the agent performing the task. The following is not intended to be an exhaustive list of battlefield tasks, but is a presentation of those tasks that are most pertinent to the decision making model.

1. Terrain Assessment

Soldiers at all levels must be able to use terrain to their advantage. To do this they must have the ability to detect terrain features and identify their characteristics. Specifically they must be able to identify terrain that could be an obstacle to movement and/or barrier to observation. Soldiers must be able to do this by directly observing the terrain in their immediate vicinity and by studying terrain representations, such as maps, to determine the nature of distant terrain. These skills are particularly important for route planning.

2. Target Assessment

Once an enemy element has been identified, leaders analyze it to determine its disposition and intentions. Targets must be prioritized based on the threat that they present and their overall potential to disrupt the mission. This involves assessing the target's position, heading, speed, capabilities, and intentions.

3. Target Acquisition

Once a target has been identified and assigned to a specific subordinate element the responsible element must acquire the target. This involves first locating where the target is with respect to the friendly element's position and then determining how to manipulate the weapons system to aim at the target.

4. Fire Control

Fire control refers to both controlling the rate of fire and the distribution of fire if multiple targets are assigned. Once a target has been detected, assigned, and acquired, the crew commander must determine when and if to fire and in what sequence to engage the targets if there is more than one. In making the firing decision the crew commander must consider factors such as expected probability of hit, ammunition status, sight alignment, actions of adjacent units, threat presented by the target, and mission guidance. If multiple targets are present, the crew commander

must determine the relative priority of the targets and the sequence of engagement.

5. Concept of Operations Development

At the unit level, commanders assess the overall situation and make decisions about the best way to allocate their assets for a given situation. This process involves an analysis of the unit's mission and an assessment of its resources. Additionally, this process must analyze the enemy capabilities and intentions and the characteristics of the terrain that the unit will operate in. From this process a plan is developed that prescribes how the unit will organize its forces and what specific missions will be assigned to subordinate elements.

6. Coordination and Communication

Unit leaders coordinate the actions of their subordinate elements to ensure that all individual actions work toward the accomplishment of the unit's overall mission. To perform this coordination, leaders closely monitor the situation and make adjustments to their plans as required. Information that could potentially impact on other elements within the unit must be made readily available to those elements.

D. NPSNET AF DECISION MAKING MODULE ARCHITECTURE

This section presents the top level architecture designed to address the battlefield decision making tasks discussed in

the previous section. The NPSNET AF decision making module is divided into three sub-modules corresponding to the tactical decision making functional areas presented earlier. The sub-modules are; command and control, movement and route planning, and target engagement. Figure 1 depicts the AF decision making module components and their relationship. These three sub-modules are described in detail in Chapters III, IV, and V of this thesis.

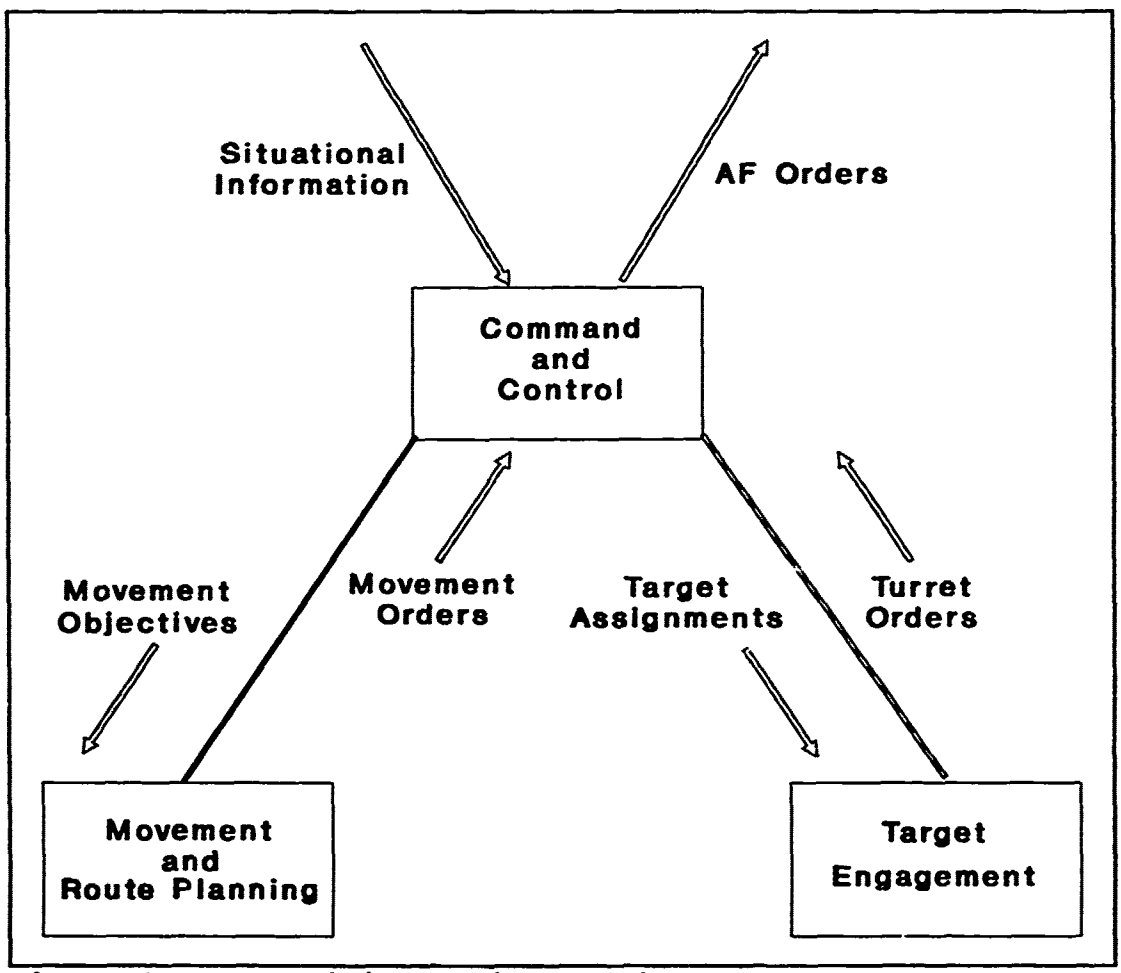

Figure 1: AF Decision Making Module

III. AF COMMAND AND CONTROL

This chapter describes the command and control module of the AF program. Command and control procedures perform the high level planning and coordination functions that generate, direct, and monitor a combat force's actions. Command and control functions occur at each organizational echelon within a unit. The command and control module of the AF program develops the AF's plan of action, monitors the situation during execution of this plan, and modifies the plan as required. It decides the AF's scheme of maneuver as a function of the initial mission, tactical principles, and conditions in the environment.

It is important to note the level of detail of the output from the command and control module. The command and control module does not generate a sequence of specific steps for each vehicle to follow. Command and control output takes the form of general mission-oriented goals for each sub-element based on the current situation.

Specifically, the command and control module performs the following functions, which are discussed in detail in the rest of this chapter.

- Tactical concept development
- Assignment of tasks to subordinate units
- Target assessment and assignment

- Communication

A. TACTICAL CONCEPT DEVELOPMENT

Tactical concept development encompasses all of the actions and decisions that determine the AF's scheme of maneuver for a given situation. This process is first performed during initialization to generate the initial plan of action for the AF. Subsequently, it is performed continuously during execution to modify the plan as conditions warrant. Concept development involves the analysis of overall mission guidance, enemy capabilities and intentions, friendly capabilities, and terrain in the area of operations. This process determines how the AF will allocate its assets to perform its assigned mission. The product of this process is a set of goals that, once achieved, will contribute to the accomplishment of the AF's overall mission. These goals are usually terrain oriented but could also be oriented on enemy forces. Terrain oriented goals could either be march objectives for subordinate elements to move to and occupy or they could be specific areas to which the enemy is to be denied access. Enemy oriented goals are specific enemy locations that the user provides the AF at initialization or that the AF detects during execution. Concept development is a function of the areas described in the following sections.

1. Mission Guidance

The AF can perform a variety of missions during the course of execution. It receives its initial mission guidance during system initialization. During mission planning a combat unit receives guidance about how it is to conduct an operation. This guidance includes a statement of the type of mission it will execute, key locations (such as starting points and final march objectives), and instructions dictating the conditions under which enemy forces can be engaged. Similarly the AF is given mission guidance during program initialization. This guidance is provided in the form of parameters that are passed during initialization. AF mission guidance parameters are listed below.

- Type of mission (e.g., offensive or reconnaissance)
- Starting location
- Initial march objectives
- Maximum range at which to consider a target a threat
- Minimum acceptable probability of hit when engaging a target

These parameters allow the user of the system to control the operational characteristics of the AF in order to ensure that his training objectives are met.

2. Enemy Forces

The AF has access to the data structures that prescribe the characteristics of all enemy forces and weapons

systems in the simulator. These data structures provide information such as armament, speed, and armor protection. This is the equivalent of the general knowledge that military forces have about their opponent's equipment. The AF uses this information and its knowledge of enemy locations to develop its tactical concept.

3. Friendly Forces

During initialization the AF is passed parameters that prescribe the characteristics of the friendly weapons systems. These parameters specify what type of unit the AF is (tank, infantry, etc.) and certain performance characteristics that dictate the AF's skill level and how it will operate. These characteristics include the following items.

- Maximum speed
- Maximum turning rate
- Maximum turret rotation rate
- Fuel and ammunition capacities
- Fuel consumption rate
- Weapons accuracy constant

4. Terrain

A high-level terrain analysis is conducted during concept development to assess its impact on the operation. This analysis focuses on the identification of large obstacles to movement such as mountains, lakes, or thick forests. Individual, isolated obstacles to movement such as trees or

shell craters are not considered at this level. During terrain analysis the AF attempts to identify the major movement corridors in the area of operations and any key or dominating terrain features. Tentative march or defensive objectives are examined to ensure that they are not in untrafficable terrain.

B. ASSIGNMENT OF TASKS TO SUBORDINATE UNITS

Once the tactical concept has been developed, specific AF elements are allocated against specific objectives identified during concept development. This involves an assessment of each AF element's current situation to determine which element should be assigned to a specific objective. To do this, the AF considers the distance from a given AF element to the objective, whether the AF element is currently in contact with the enemy, and the relative priority of the AF element's current mission assignment. AF element mission assignments fall into one of three categories. These categories are listed below in descending order of priority.

- Attack of a high threat target
- Reinforcement of another AF element with multiple high threat targets
- Movement to assigned march objective

Target threat levels are described in a later section. The reinforcement mission is assigned when one AF element is engaged with more high threat targets than it can effectively

handle. In this situation an uncommitted element is dispatched to provide assistance to the over-committed element. The mission assignment process is shown below.

- Identify the priority of the new mission
- Determine which AF element is closest to the new mission objective
- If the closest element's current mission is a lower priority than the new mission, then the new mission is assigned to that element
- If the closest element's current mission is a higher priority than the new mission, then repeat the above process for the next closest element
- If the current mission for a given AF element and the new mission are of equal priority and no other AF element with a lower priority mission exists, then assign the AF element to the mission objective closest to its current position

Figure 2 depicts an example of a mission assignment scenario. In this example platoon 1 is pursuing a priority 2 mission objective, platoon 2 is pursuing a priority 1 objective, and the AF command and control module has identified a new mission objective. Although platoon 2 is the closest element to the new objective, it will continue to pursue its existing objective since it is of higher priority. Platoon 1's existing objective is of equal priority to the new objective but is more distant, therefore the new objective is assigned to platoon 1.

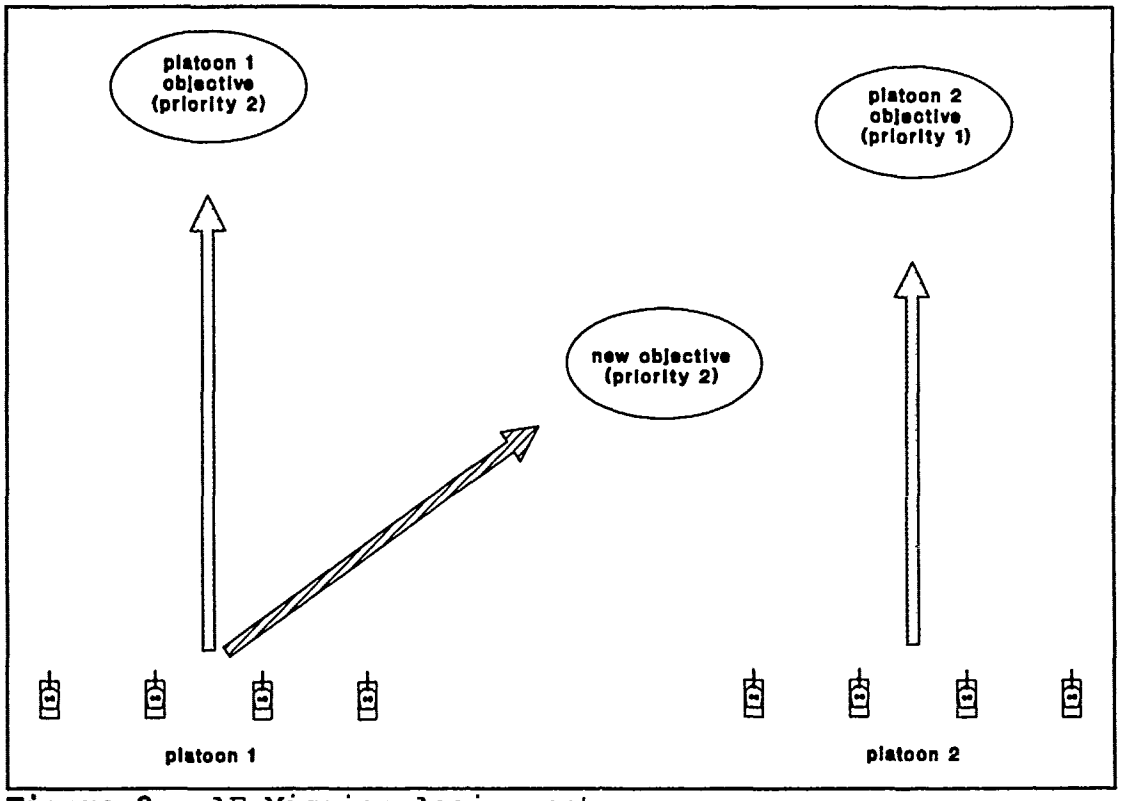

Figure 2: AF Mission Assignment

C. TARGET ASSESSMENT AND ASSIGNMENT

Once a target has been detected it is analyzed to determine the threat it presents to the AF mission. This analysis involves an assessment of the target's location, firepower, mobility, armor protection, and intentions. For example, close targets are a higher threat than distant targets, armored vehicles are a higher threat than non-armored vehicles, and attacking targets are a higher threat than retreating targets.

Target assignment decisions are made at two levels. First, at the top level a newly identified target is analyzed to determine which, if any, AF platoon it will be assigned to.

Each AF platoon can be assigned up to four targets simultaneously. Next, at the platoon level the assigned targets are analyzed to determine which individual tank(s) will have responsibility for the target. The target assignment process at these two levels is described in the following sections.

1. Platoon Target Assignment

The range to a target from each AF platoon coupled with any existing target responsibilities determine if any of the AF platoons will be assigned responsibility for a new target. The range to a target is classified in one of three categories--high threat, medium threat, or low threat. The thresholds for these categories are set during initialization. If the target is classified as low threat, it will not be considered further during that decision cycle. If the target is a medium threat, then the responsible AF platoon will be directed to orient its weapons on the target but continue toward its assigned march objective. If the target is a high threat to an AF platoon and no other target presents a greater threat, then the AF platoon will be directed to attack the target. When this occurs the AF platoon places its original mission on hold and orients its weapons and movement on the assigned target. The AF element will pursue the target until it has destroyed it or until the target has moved far enough

away to no longer be considered a high threat. The platoon target assignment process is shown below.

- Calculate ranges from each platoon to each target
- Remove low threat targets
- If a platoon has more than four targets, remove the least threatening targets
- Assign each remaining target to the closest platoon
- If one or more platoon target is a high threat, assign the closest target as an attack objective for that platoon

2. Individual Tank Target Assignment

Once targets have been allocated to each platoon, the decision must be made which tank(s) within the platoon will have responsibility for each target. If the platoon has only one target assigned, then all tanks will orient on that target. However, if there are multiple targets assigned to the platoon, the platoon's fires must be distributed among them. An outside to inside fire distribution technique is used. The left-most and right-most targets from the platoon's perspective are identified and assigned to the left-most and right-most tanks within the platoon. If there are more than two targets the process is repeated until all targets are assigned to a specific tank. Figure 3 depicts an AF platoon distributing fires among two targets. Appendix B presents a detailed description of the fire distribution process.

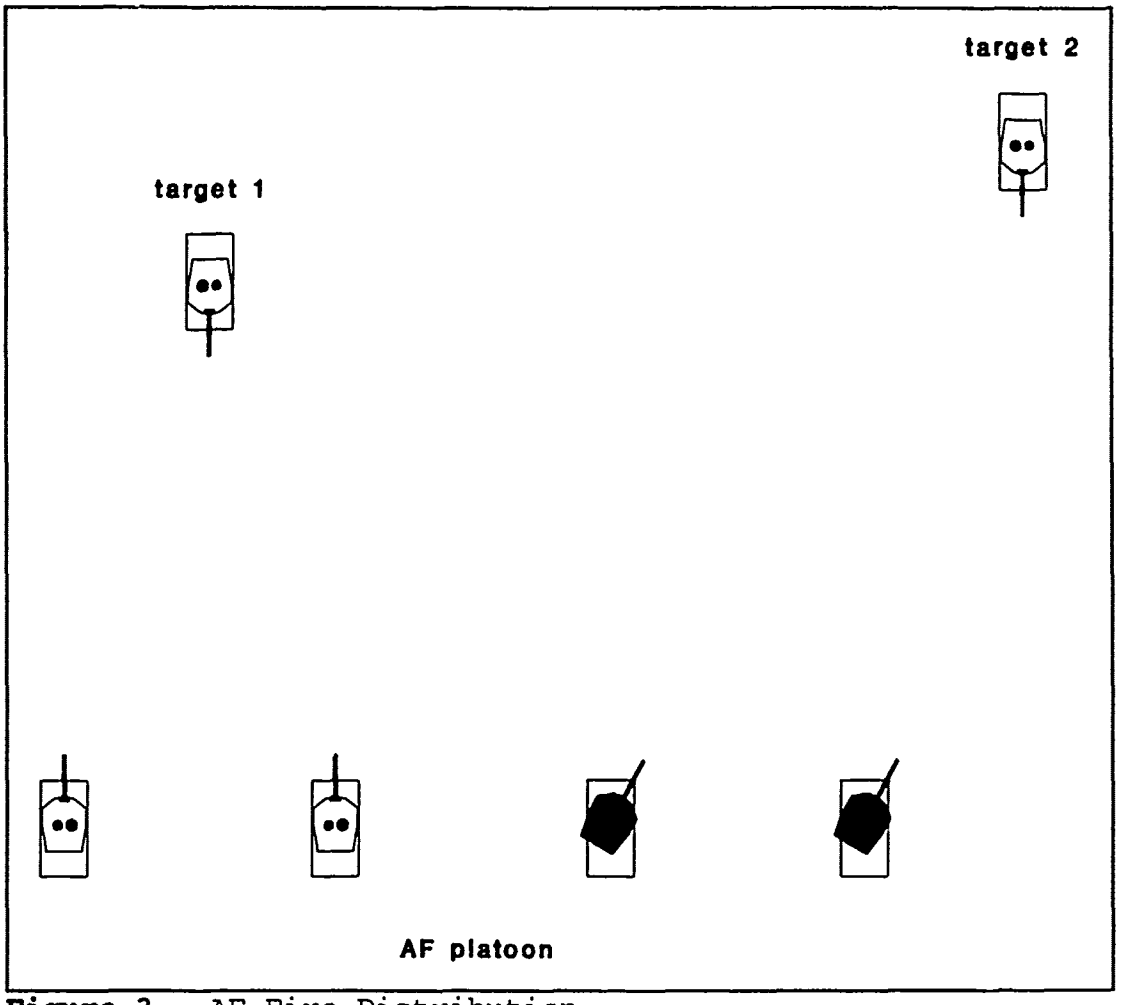

Figure 3: AF Fire Distribution

D. COMMUNICATION

The AF must be able to perform as a team rather than as several independently operating vehicles. This is accomplished with procedures that make information known by one element also available to other elements as appropriate. This communication is effected by making pertinent information global in nature so that it can be accessed by other AF elements and the command and control functions. The

communication of mission taskings is accomplished through the assertion of weapons orientation goals and vehicle position and orientation goals.

IV. AF MOVEMENT AND ROUTE PLANNING

This chapter describes the techniques employed to generate movement orders for the AF. The movement decision routines presented here are at a lower decision making level than the command and control routines described in Chapter III. Given a march objective--generated by the command and control module--the movement module determines the actions necessary to move the vehicles toward the objective. Movement decisions occur in three phases. A path to the assigned march objective is first developed (route planning). Next, platoon movement decisions are made to keep the platoon oriented on the selected path. Finally, individual vehicle movement decisions are made to keep the vehicles in the proper position within the platoon formation. Each movement decision must account for both speed and directional components. The following sections describe the three phases of movement order generation.

A. ROUTE PLANNING

AF route planning is conducted to ensure obstacle avoidance and realistic tactical use of the terrain. This involves the consideration of the platoon's current location, the location of its march objective, and the intervening terrain and obstacles. The path generated is expressed as a

series of intermediate movement objectives for the AF platoon. Path generation techniques fall into one of two general categories--incremental path planning and *a priori* path planning [Ref. 11]. These two techniques are briefly described here.

The *a priori* approach involves the calculation of a movement network through the world. The starting location, final march objective, and all obstacles are represented as nodes in the network. The network arcs represent all possible route segments. Using a shortest path algorithm, the optimal route between any two points in the world is generated. The initial network is generated off-line and is accessed each time a path is assigned. This approach will always return the optimal path between any two points but requires significant memory space to store the world network and can increase overall decision making time. Additionally, the *a priori* approach does not address dynamic obstacles or allow for the dynamic assignment of march objectives outside of those used in the initial network generation.

The incremental approach involves dynamic route planning as the AF moves through the world. In this approach, the AF examines the terrain for a limited distance in the direction of the assigned march objective and selects the best path based on the limited area that it considers. As the AF progresses toward its march objective, new terrain is considered and the path is updated. The incremental approach

may not select the optimal path to the final march objective, but it is computationally faster and less complex than the *a priori* approach and can handle dynamic situations.

An alternative approach is to use a hybrid of the a priori approach. An initial network is generated using the a priori approach and is used as described above. During execution, as new march objectives are generated, the network is updated to include the current position and new march objective as additional nodes. The shortest path algorithm is then recomputed. This approach allows the selection of optimal routes and addresses operations in a dynamic environment.

B. PLATOON MOVEMENT

During each decision cycle a platoon's location and heading are evaluated based on its assigned march objective. If a platoon is not properly oriented on its goal, then the platoon movement routine will calculate the necessary movement corrections. Platoon positions are calculated and monitored based on an imaginary point at the center of the platoon formation. This point is referred to as the base point. Platoon movements are constrained to ensure that the platoon members can maintain their formation positions. The platoon movement decision involves first the calculation of the turn required to achieve the desired orientation and then the constraint of this turn if it exceeds the physical limits of the platoon members' turning rate or maximum speed.

The turning limit of a platoon is a function of the maximum distance the vehicle on the outside of the turn can travel during a given time period. If the platoon cannot make the desired turn, then the maximum possible turn in the desired direction will be executed. The average decision cycle time is used to estimate how far a platoon can turn during a given cycle. For example, if the average decision cycle time is one second and the outside vehicle is moving at ten meters per second, then for planning purposes the platoon cannot execute any turn requiring the outside vehicle to travel more than ten meters. The average cycle time is used throughout the model for planning and to realistically constrain time dependent AF activities.

The movement procedures described in this thesis apply to a platoon in a line formation, i.e., all four vehicles travelling abreast of one another at 50 meter intervals. Variations on these procedures can be used for different platoon formations. Figure 4 depicts a platoon turn. Appendix C provides a detailed presentation of the platoon movement calculations.

C. INDIVIDUAL VEHICLE MOVEMENT

Once a platoon movement order has been generated, the move required by each platoon member can be calculated. The new platoon heading and base point dictate how each platoon member will move in order to remain in formation. Each movement

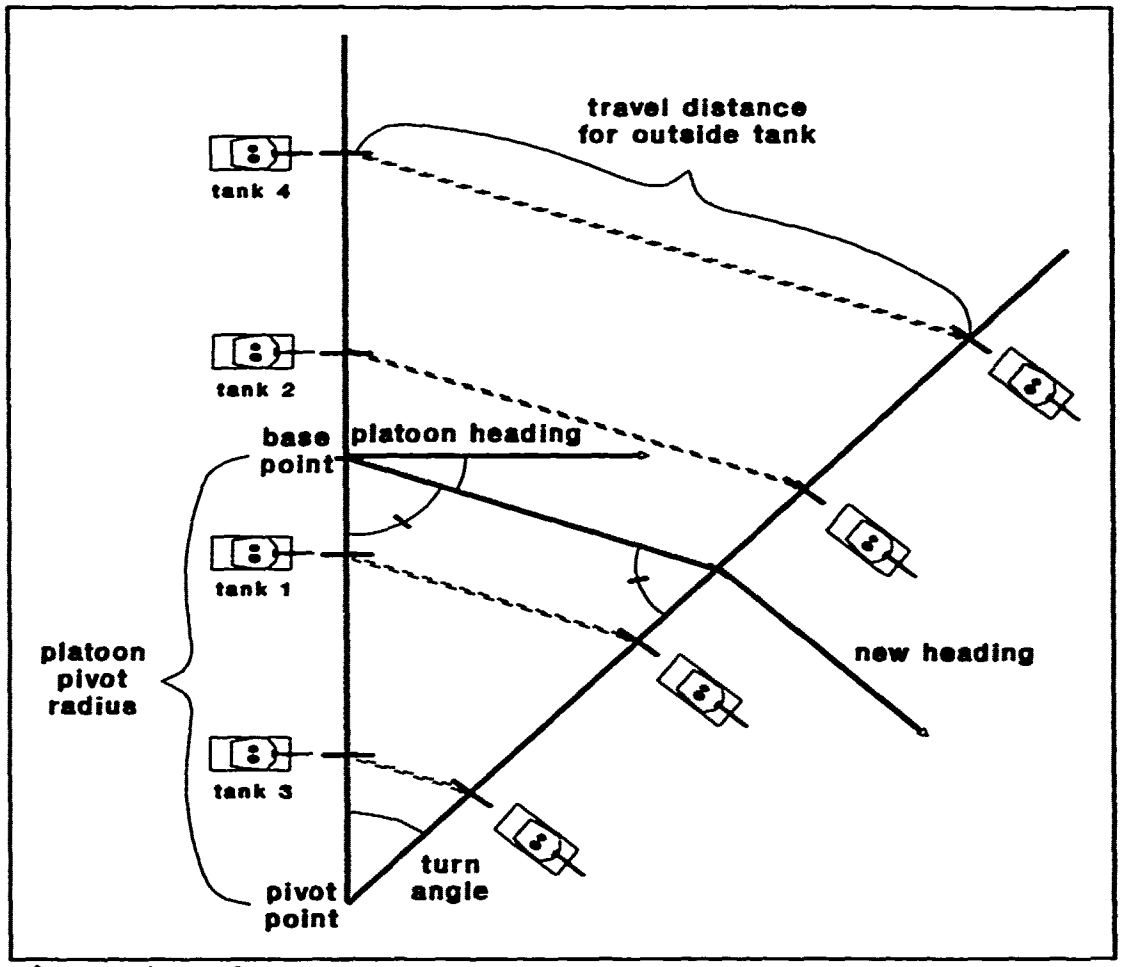

Figure 4: Platoon Turn

formation will require its own routine for generating formation positions. This discussion will continue based on the platoon line formation. Each vehicle in a platoon is assigned a position number from one to four. From left to right the positions are four, two, one, three. Placing the even numbers on one side and the odd on the other side facilitates the referencing of vehicles based on the side of the platoon that they are on. The position number describes where the vehicle is in the formation. The coordinates of a

vehicle's formation position are a function of an offset direction and distance from the platoon base point. A vehicle movement order consists of the heading and speed required to reach the next formation position. Appendix C provides a detailed presentation of the vehicle movement calculations.

V. TARGET ENGAGEMENT

This chapter discusses the AF program target engagement process. This process involves sector scanning, target acquisition, and fire control procedures. Sector scanning refers to the observation of an assigned sector for enemy activity. Target acquisition covers the mechanics of properly aligning a target in the weapon's sights. Fire control pertains to the process of determining if and when to fire at an acquired target.

A. SECTOR SCANNING

In the absence of an assigned target each tank will orient its turret on a designated sector and "watch" for targets. In a platoon formation the left-most and right-most tanks will observe to the platoon's left and right respectively. The inner tanks will observe directly to the platoon's front. When there are no targets assigned, the left-most tank will automatically shift to a turret position of 315 degrees, the right-most tank to a position of 45 degrees, and the inner tanks to a position of zero degrees.

B. TARGET ACQUISITION

Once a target assignment has been generated by the command and control process, the target's position, range, and speed

are made available to the responsible tank. The assigned target must first be located with respect to the responsible tank's position in order for the required aiming actions to be calculated. The order to fire at a target is given as a part of the overall tank order generated during the decision cycle. When the decision to fire is made, it is executed after the chassis move that is included in the same order message. Therefore, all aiming calculations are based on the next chassis orientation to be implemented. The aiming process requires calculation of the desired main gun directional position and the desired main gun elevation. Appendix D provides a detailed description of the aiming process.

C. FIRE CONTROL

This section describes the decision factors used to determine when to fire the main gun. Upon receiving a target assignment, a determination is made if the conditions are right to issue the fire order. The decision is based on the accuracy of the main gun's target alignment, the probability of hit for the current conditions, and on a verification that the tank has a clear line of sight to the target. The firing decision routine checks these criteria in order of complexity beginning with the simplest. Each criterion must be satisfied before the next criterion will be considered. The basic firing decision process is shown below.

- If main gun is correctly aligned on the target, then calculate probability of hit
- If probability of hit is acceptable, then check for line of sight to target
- If line of sight to target exists, then issue fire order

These criteria are discussed in detail in the following sections.

1. Main Gun Alignment

The desired main gun direction and elevation calculated during target acquisition are compared to the current main gun alignment. If the current main gun orientation equals the desired orientation then the main gun alignment requirement has been met. If the current orientation does not equal the desired orientation then a "no fire" decision is made without consideration of any other criterion. If the alignment requirement is met the firing decision process will continue.

2. Hit Probability Assessment

With the knowledge that the main gun is properly aligned, an assessment is then made of the expected probability of hit for the given conditions. The hit probability is a function of the range to the target, the speed of the firing vehicle, the speed of the target, and a weapons accuracy constant for the firing vehicle [Ref. 12]. The probability is calculated as follows.

s_{tank} = *firing vehicle's speed* r = *range to target*
s_{tgt} = *target's speed* k = *weapon accuracy constant*

$$Probability\ of\ hit = (100 * e^{k * r^2}) - \frac{s_{tank}}{2} - s_{tgt}$$

If the probability of hit is greater than or equal to a minimum value specified during initialization, then the hit probability criterion is satisfied and the firing decision process continues.

3. Line of Sight Check

A line of sight check is made to determine if the line of fire to the target is free from obstruction. In reality the line of sight check occurs without any thought being given to it. If the gunner can see the target, then he knows he has line of sight. In a computer program, it is not so simple and in fact is the most time consuming of the firing decision routines. Therefore, it is only performed if all other firing criteria are satisfied. The line of sight check first determines if other friendly tanks are in the line of fire. This is accomplished by comparing the direction to the target with the direction to the other friendly vehicles. If the target direction matches the direction to a friendly vehicle and the friendly vehicle is closer than the target, the line of sight check will fail. If there are no friendly tanks in the line of fire, the routine checks for terrain features that block the line of sight to the target. The line of sight

terrain check calculations are presented in Appendix D. This check is the final firing decision criterion. Once this criterion has been satisfied a fire order will be issued.

VI. IMPLEMENTATION

The Autonomous Force model as presented in this thesis has been partially implemented in the NPSNET simulation system. This chapter describes the current AF model prototype. This implementation incorporated the belief generation module presented in [Ref. 5] and the decision making module presented in this thesis into a single run-time program. The program is a combination of procedural routines and embedded rule-based modules. The AF program currently runs on a Silicon Graphics IRIS workstation and communicates with the NPSNET simulator via Ethernet.

A. PROGRAM DESCRIPTION

The AF program is written in the 'C' programming language with embedded rule-based modules written in the 'C' Language Integrated Production System (CLIPS). 'C' was selected for the host program to facilitate interaction with the simulator routines which are also written in 'C'. Additionally, the use of 'C' greatly simplifies the integration of CLIPS modules since CLIPS was specifically designed to work with 'C'.

The 'C' host program performs the low level procedural functions of the AF program. These include general housekeeping, network communication, and program control

functions. It also maintains the global data structures representing the state of the world.

The CLIPS modules are the heart of the AF program. They perform the reasoning and decision making functions for the AF. The CLIPS modules were developed separately, converted to 'C' code, and then compiled as part of the overall AF run-time program. [Ref. 13] provides a complete description of the CLIPS to 'C' conversion process. There are three embedded CLIPS modules called by the host program. Each of these will be briefly described in the following sections.

1. AF Initialization Module

This module is called once at the beginning of an AF run. It includes a simple user interface to allow the user of the system to prescribe the starting AF conditions, mission, and certain behavioral characteristics. Based on the user's input, the initialization module develops the initial set of AF orders.

2. Belief Generation Module

This module is called during each execution cycle and generates the perceived state of the world for the decision making module to use. The belief generation module introduces a degree of error to the factual information it receives in order to model the limitations of sensors and human perception. The user can control the skill of the AF by

varying the level of error introduced by the belief generation module.

3. Tactical Decision Making Module

This module is also called during each decision cycle. It takes the believed information generated by the above module and decides what actions the AF will take based on this information. The actions described in Chapters III-V of this thesis are performed here. This module generates the AF order messages that are sent to the simulator.

B. EXECUTION CYCLE

Once initialization is complete the AF program loops through its execution cycle until terminated by the user. The basic execution cycle for the AF program is as follows.

- Read update messages from network
- Convert world information to beliefs
- Update data structures
- Generate tactical decisions
- Send AF orders to simulator

1. Read Update Messages from Network

The NPSNET simulator generates update messages each time the state of a vehicle changes. The vehicle state consists of vehicle heading, speed, gun direction, firing status, and alive or dead status. The update messages are sent over Ethernet and placed in a message buffer at each

station on the network. At the start of the execution cycle, the messages in the buffer are read by the AF program and the new vehicle states are stored in data structures for later use.

2. Convert World Information to Beliefs

The belief generation module is called here and performs the conversion of factual world information to believed information. This module processes all of the factual information about user driven vehicles and places the resulting believed information in a separate data structure for use by the decision making module.

3. Update Data Structures

As mentioned earlier, network update messages are only generated when a vehicle's state changes. AF vehicle state messages, received from the simulator, are directly stored as updated information in the program's global data structures. Updated believed states, are stored at the conclusion of the belief generation routine. It is the responsibility of each station on the network to dead reckon the location of each vehicle that is not included in an update message. This process occurs once each execution cycle immediately prior to the decision making routine. New positions for all vehicles being tracked by the program are calculated based on the last location, speed, direction, and time.

4. Generate Tactical Decisions

The decision making module is called here. The current AF and belief information are asserted to the CLIPS fact list and are the basis for the next round of orders. The decision making module considers the current situation and decides what actions to take. If an action involves changing the state of an AF vehicle, then the module calls an external function to send the new AF vehicle state over the network. At the conclusion of the decision making routine, all situational information such as AF missions are stored in the external global data structures.

5. Send AF Orders to Simulator

Any AF vehicle state changed by the decision making module will result in an update message being sent to the simulator. This function takes the information as used by the decision making module and converts it to the format required by NPSNET. The reformatted information is then transmitted over the network.

C. IMPLEMENTATION ASSESSMENT

The incorporation of the belief generation and decision making functions into the same program greatly facilitated the passing of information between these two modules. The use of the 'C'--CLIPS approach provided many advantages during development. The primary advantage was the ability to fine tune the CLIPS modules during execution. Prior to generating

the run-time program, the 'C' host program was created with calls to load the CLIPS modules from files and run them during each execution cycle. While this file I/O slowed performance it had the advantage of allowing the modification of the CLIPS routines during program execution. This provides immediate feedback on a modification since changes made in this manner are reflected during the next decision cycle. Additionally, because the CLIPS routines were being loaded from a file, there was no requirement to recompile the program except when the host program itself was changed. For ease of program development and rapid testing of ideas, this was an ideal solution.

The AF run-time program currently runs at a marginally acceptable speed to keep pace with operations in the simulator. The time for each execution cycle averages around two seconds during the early part of a program run. The use of rule-based intelligence in the program results in high memory utilization. As time progresses the average execution time begins to slow down as memory usage increases. AF behavior begins to be adversely affected when the time for an execution cycle exceeds around five seconds. The AF program currently runs for around 1000 decisions cycles before memory utilization becomes a problem. Part of this problem is due to the lack of an IRIS specific CLIPS implementation. The UNIX System V CLIPS implementation does not work properly on the IRIS system. Because of this, the AF model uses a generic

CLIPS implementation that does not optimize memory utilization. The execution cycle time will become more of a problem as higher levels of sophistication are added to the decision making process.

VII. CONCLUSIONS AND RECOMMENDATIONS

This chapter evaluates the AF model described in this thesis and proposes areas for future work. The AF model presented here constitutes a viable framework for an autonomous force in a combat simulator at the level of sophistication described in Chapters III-V. Though this implementation does not have full functionality, the results achieved thus far by this model support the validity of this approach.

A. CONTRIBUTIONS OF THIS RESEARCH

The most significant contribution of this thesis is the development of a conceptual framework upon which a fully functional autonomous force can be built. This research identified and defined the functional requirements of the NPSNET Autonomous Force. These requirements were translated into a computational model and system architecture for the AF. The thesis resulted in a working prototype for the AF and conducted limited testing and experimentation with the model.

The NPSNET Autonomous Force contributes significantly to the realism and training effectiveness of the NPSNET simulator. Users of the system now face a challenging and reasonably realistic opposing force as they maneuver on the simulated battlefield. This is achieved without any

additional personnel overhead, i.e., there is no 'man in the loop' in NPSNET opposing force operations. Additionally, the AF model allows simulator users to tailor the opposing force's skill level, behavior, and capabilities to support their specific training objectives.

B. ASSESSMENT OF CURRENT AF MODEL

The AF model, as currently implemented, performs a subset of the functionality described in this thesis. Initial results from AF simulator runs were very encouraging. Limited testing against human opponents indicates that the AF provides a reasonable level of realism for the missions that it currently conducts. The AF is able to make expert-level movement and target engagement decisions. In these areas the AF has outperformed human players during test runs. The basic command and control functions of target assignment, subordinate element mission assignment, and coordination of vehicles at the platoon level perform acceptably well for a first implementation. The coordination of platoons at the company level is currently performed using simplified algorithms and requires further development to achieve expert-level performance. The route planning and obstacle avoidance routines have yet to be implemented and are the most significant shortfall in basic AF operations. Additionally, the top level strategy making functions are not in place to allow the AF to conduct coherent campaigns.

C. RECOMMENDED FOLLOW ON WORK

This section provides recommendations for the next phase in NPSNET AF development. The recommendations cover actions to improve AF behavior and steps to improve AF program performance.

While this thesis makes a significant contribution to the development of NPSNET autonomous forces, several areas must be addressed in order to elevate the AF to true human-like behavior. The area most in need of immediate work is the implementation of route planning and obstacle avoidance procedures. Procedures must be developed that allow the AF to detect and avoid trees, buildings, and other obstacles, and to make proper tactical use of terrain during movement and engagements. Next, command and control procedures to coordinate platoon actions need to be improved. Optimization routines for the allocation of the AF platoons against multiple mission requirements should be incorporated into the program. Currently, the mission assignment and fire distribution routines produce a workable solution but not necessarily the optimal solution. Finally, at the highest decision making level, the AF needs the ability to develop and coordinate the execution of an overall campaign plan. Currently the program views all activities that are detected as isolated events and does not have the ability to piece together the big picture of what is going on in the world.

As alluded to in Chapter VI, the AF program may already be approaching the performance limits for the current design and level of sophistication. Therefore, before substantial improvements can be made to the decision making procedures, performance improvements will have to be made. There are several ways in which improvements could be made. First, the performance of the current program could be improved by implementing it on a platform with a system specific CLIPS implementation, such as on the Sun workstations. This will result in more efficient use of memory and therefore faster execution speed. However, this will require the integration of the new system into the NPSNET network. Second, computational efficiency was not a priority in the development of the initial AF implementation, so significant improvements in execution speed could be achieved through a careful optimization of the program. Finally, the nature of the program seems to be well suited to parallel processing. Once the command and control functions have been performed, the remainder of the decision making program involves multiple iterations of platoon and individual vehicle decisions. Significant performance improvement would result if platoon and individual vehicle level decisions could be assigned to separate machines and processed in parallel.

This thesis is a significant first step in the development of a fully functional autonomous force for NPSNET. While our

prototype is only a partial implementation, it represents a solid foundation for the next phase of AF development.

APPENDIX A - NPSNET/AF REPRESENTATION CONVENTIONS

1. NPSNET Coordinate System

World locations are expressed in terms of a three dimensional cartesian coordinate system with the origin located at the southwest corner. The positive x-axis runs east, the positive z-axis runs north, and the positive y-axis represents elevation. Directional angles are expressed using standard compass heading conventions. Due north is represented as zero degrees and the value of a directional angle increases as it rotates clockwise toward the x-axis. This is different from normal mathematical conventions and is accounted for by manipulating the use of the trigonometric functions and making appropriate adjustments for sign.

2. Tank Main Gun Orientation Conventions

Tanks are turreted vehicles and as such have multiple degrees of movement associated with the aiming of their main gun. A tank turret can rotate in 360 degrees and the main gun can be elevated or depressed to certain vehicle dependent limits. The main gun orientation is expressed with respect to the vehicle chassis. The chassis orientation must therefore be known before the main gun can be aimed.

Main gun directional orientation is described as an offset to the longitudinal axis of the chassis. The turret position

is expressed as a clockwise rotation (looking down on the vehicle) from 0 to 360 degrees from the front of the vehicle. For example, a turret position of zero degrees indicates the main gun is oriented directly to the front of the vehicle; a position of 90 degrees indicates the main gun is oriented directly off the right side of the vehicle.

The main gun's vertical orientation is described as an offset to the horizontal plane of the chassis. Main gun elevation is expressed as an upward rotation from zero degrees to the upper limit of the weapon system and from 360 degrees downward to the lower limit. For example, an elevation of zero degrees is parallel to the chassis, 15 degrees is an upward elevation of 15 degrees, and 345 degrees is a downward depression of 15 degrees.

APPENDIX B - FIRE DISTRIBUTION CALCULATIONS

The fire distribution process begins by calculating the directions from a platoon to two targets under consideration (eq B.1) (see Appendix A for a description of the NPSNET coordinate system). Next, the angles between the platoon heading and the target directions, called the target-angles, (see Figure 5) are calculated (eq B.2). The target-angles are examined to determine which target is the left-most and right-most. In order to account for the effects of a target angle that brackets the zero degree line, there are two cases that must be considered. First, if both target-angles bracket the zero degree line or both do not, then the smaller of the target-angles corresponds to the right-most target (eq B.3). In the second case, where one of the target-angles brackets the zero degree line but the other one does not, the smaller of the target-angles corresponds to the left-most target (eq B.4). Having identified the left-most and right-most targets, they are then assigned to the corresponding tanks in the platoon formation. The fire distribution process is formally stated below.

θ_{t1} = *direction to tgt* 1 $\quad$ $\angle_{ta1}$ = *target angle* 1

θ_{t2} = *direction to tgt* 2 $\quad$ $\angle_{ta2}$ = *target angle* 2

x_{t1}, z_{t1} = *tgt* 1 *location* $\quad$ x_{t2}, z_{t2} = *tgt* 2 *location*

x_{plt}, z_{plt} = *platoon location* $\quad$ θ_{plt} = *platoon heading*

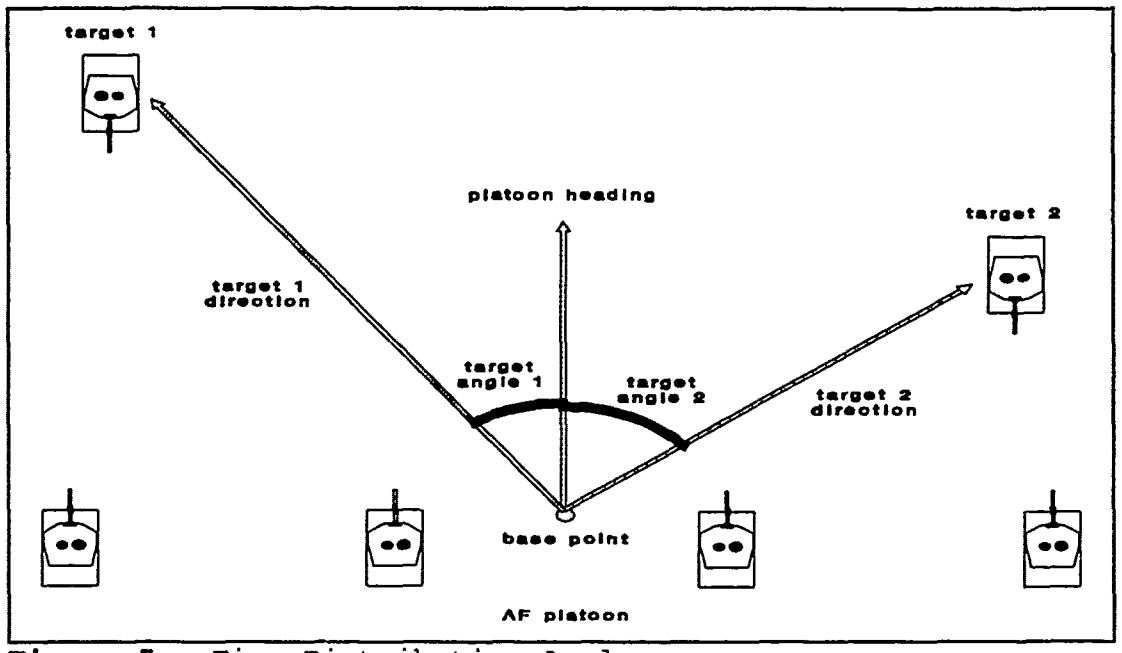

Figure 5: Fire Distribution Angles

$$(B.1)\quad \theta_{t1} = (\arctan\frac{x_{t1} - x_{plt}}{z_{t1} - z_{plt}} + 360°) \bmod 360$$
$$\theta_{t2} = (\arctan\frac{x_{t2} - x_{plt}}{z_{t2} - z_{plt}} + 360°) \bmod 360$$

$$(B.2)\quad \angle_{ta1} = \theta_{plt} - \theta_{t1}$$
$$\angle_{ta2} = \theta_{plt} - \theta_{t2}$$

$$(B.3)\quad \textit{if } [(|\angle_{ta1}| \leq 180°) \wedge (|\angle_{ta2}| \leq 180°)] \vee [(|\angle_{ta1}| > 180°) \wedge (|\angle_{ta2}| > 180°)]$$
then if $\angle_{ta1} \leq \angle_{ta2}$
then target 1 *is right and target* 2 *is left*
else target 1 *is left and target* 2 *is right*

$$(B.4)\quad \textit{if } [(|\angle_{ta1}| > 180°) \wedge (|\angle_{ta2}| \leq 180°)] \vee [(|\angle_{ta1}| \leq 180°) \wedge (|\angle_{ta2}| > 180°)]$$
then if $\angle_{ta1} < \angle_{ta2}$
then target 1 *is left and target* 2 *is right*
else target 1 *is right and target* 2 *is left*

APPENDIX C - MOVEMENT CALCULATIONS

1. Platoon Calculations

The direction to a movement goal from the platoon's position is first calculated using the same procedure as in equation B.1. Next, the value of the most direct angle between the goal direction and the platoon's current heading is calculated. This value, called the turn-angle, is the basis for the rest of the directional calculations (see Figure 6). The turn-angle is the difference between the goal direction and the current heading (eq C.1). If the goal direction and the current heading bracket the zero degree line, then the turn-angle is adjusted to account for this (eq C.2). A negative value for the turn-angle indicates a turn to the left. The unconstrained turn calculation is shown below.

θ_{plt} = *platoon heading*
θ_{goal} = *goal direction*
$\angle_{ta}$ = *turn-angle*

(C.1) $\angle_{ta} = \theta_{goal} - \theta_{plt}$

(C.2) *if* $|\angle_{ta}| > 180°$
then if $\angle_{ta} > 0°$
then $\angle_{ta} = \angle_{ta} - 360°$
else $\angle_{ta} = \angle_{ta} + 360°$

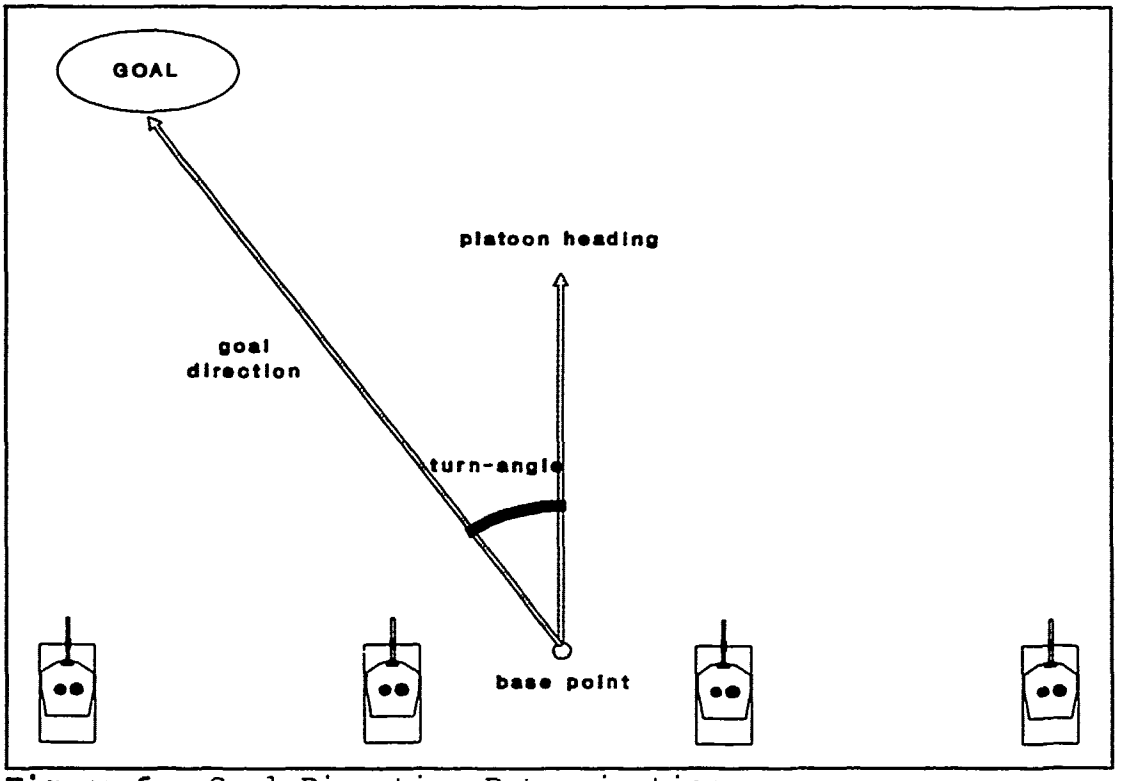

Figure 6: Goal Direction Determination

It must next be determined if the platoon could realistically make the turn calculated above during the next execution cycle. The first step in determining the platoon's turning limit is to set the platoon's new speed. If the platoon is turning, its new speed is reduced to 75% of its current speed to allow the vehicle on the outside of the turn to maintain its formation position. If the platoon is not turning, the new speed is set to the maximum platoon speed for the given situation (eq C.3). Next, the maximum distance that the vehicle on the outside of the turn can travel during the next execution cycle is calculated (eq C.4). The turn is calculated based on a pivot point 25 meters inside the inner-

most vehicle. The maximum turn that the platoon can make is equal to the central angle of the circle sector with sides equal to the pivot radius and base equal to the maximum distance the outside vehicle can travel (eq C.5) (see Figure 4). If the magnitude of the turn-angle, calculated in the previous section, is greater than the maximum possible turn, then the turn-angle is set equal to the maximum turn value but retains its sign (eq C.6). The new platoon heading is then calculated by adding the current heading to turn-angle (eq C.7). The next step is to project the base point of the platoon to its next location. This new location will provide the base from which all vehicles will offset their movement in order to remain in formation. The turn constraint calculations are shown below.

t_{avg} = *average cycle time* $\quad \theta_{plt}$ = *platoon heading*
d_{max} = *max travel distance* $\quad \theta_{goal}$ = *goal direction*
d_{plt} = *plt travel distance* $\quad \theta'_{plt}$ = *new platoon heading*
s_{max} = *max platoon speed* $\quad \angle_{ta}$ = *turn-angle*
s'_{plt} = *new platoon speed* $\quad \angle_{max}$ = *maximum turn angle*
r_{tank} = *tank pivot radius* $\quad r_{plt}$ = *platoon pivot radius*

(C.3) *if* $\angle_{ta} \neq 0$, *then* $s'_{plt} = s_{max} * .75$

$$(C.4) \quad d_{max} = s'_{plt} * t_{avg} * \frac{r_{tank}}{r_{plt}}$$

$$(C.5) \quad \angle_{max} = 2 * \arcsin \frac{d_{max}}{2 * r_{tank}}$$

(C.6) *if* $|\angle_{ta}| > \angle_{max}$
then if $\angle_{ta} < 0°$
then $\angle_{ta} = -1 * \angle_{max}$
else $\angle_{ta} = \angle_{max}$

(C.7) $\theta'_{plt} = (\theta_{plt} + \angle_{ta} + 360°) \bmod 360$

2. Individual Vehicle Calculations

The first step in generating a vehicle movement order is to determine the coordinates of the vehicle's next formation position. These coordinates are prescribed by an offset direction from the platoon heading and a distance from the next platoon base point. If the vehicle's position number is odd, the relative offset direction is 90 degrees and the offset distance is its position number multiplied by 25. If the vehicle's position number is even, the relative offset direction is 270 degrees and the offset distance is one less than its position number multiplied by 25 (eq C.8). The true direction of the offset, with respect to the world, is then calculated (eq C.9). Given this direction, the offset distance, and the coordinates of the base point, the coordinates of the vehicle's next position are calculated. These coordinates constitute a short term movement goal for the vehicle and the heading to this goal is calculated in the same manner as described in the previous section. The distance from the vehicle's current position to its movement

goal is calculated and divided by the average cycle time to determine the vehicle's new speed. The individual vehicle movement calculations are shown below.

p_n = *tank's position number* d_o = *offset distance*
θ_o = *true offset direction* s_t' = *new tank speed*
θ_{rel} = *relative offset direction*
d_t = *distance to tank's new formation position*

(C.8) *if* (p_n mod 2 = 0)
then $\theta_{rel} = 270°$
$d_o = 25 * (p_n - 1)$
else $\theta_{rel} = 90°$
$d_o = 25 * p_n$

(C.9) $\theta_o = (\theta_{rel} + \theta_{plt}) \bmod 360$

APPENDIX D - TARGET ENGAGEMENT CALCULATIONS

1. Turret Rotation Calculations

The direction from the tank's position to the target is first calculated as using the procedure from equation B.1. Next the direction of the main gun, with respect to the world, is calculated based on the next chassis move (eq D.1). Appendix A describes the conventions for expressing a tank's main gun orientation in NPSNET. The difference between the main gun direction and the target direction, referred to as the gun-target angle, provides the basis for the required turret rotation (eq D.2). Each weapons system will have a maximum turret slew rate that it can achieve. The maximum slew angle that can be achieved during the current decision cycle is calculated by multiplying the maximum slew rate by the average decision cycle time (eq D.3). If the absolute value of the gun-target angle is less than the maximum slew angle, then the turret can rotate directly to the desired position (eq D.4). Just as described in Appendix C for the movement process, turret rotation calculations must account for the case where the zero degree line is crossed over. If the magnitude of the gun-target angle is greater than the maximum slew angle, then a determination must be made if it is better to rotate the turret to the left or right (eq D.5). The turret slew calculations are formally stated below.

θ_{tur} = *turret position* θ_{tank} = *tank heading*
θ_{gun} = *gun direction* $\angle_{gt}$ = *gun-target angle*
θ_{tgt} = *target direction* t_{avg} = *average cycle time*
$\Delta_{\theta_{tur}}$ = *max slew rate* Δ_{max} = *max slew angle*
θ'_{tur} = *new turret position*

(*D*.1) $\theta_{gun} = \theta_{tank} + \theta_{tur} \bmod 360$

(*D*.2) $\angle_{gt} = \theta_{tgt} - \theta_{gun}$

(*D*.3) $\Delta_{max} = t_{avg} * \Delta_{\theta_{tur}}$

(*D*.4) *if* $(\Delta_{max} + |\angle_{gt}| + 360°) \bmod 360 \leq 2 * \Delta_{max}$
then $\theta'_{tur} = (\angle_{gt} + \theta_{tur} + 360°) \bmod 360$

(*D*.5) *else if* $((\angle_{gt} + 360°) \bmod 360) \geq 180°$
then $\theta'_{tur} = (\theta_{gun} - \Delta_{max} + 360°) \bmod 360$
else $\theta'_{tur} = (\theta_{gun} + \Delta_{max}) \bmod 360$

2. Main Gun Elevation Calculations

The first step in determining the main gun elevation is to calculate the vertical angle from the tank's position to the target (eq D.6). Next, the vertical tilt of the chassis in the direction of the target is calculated. This is accomplished by first selecting a point a short distance from the tank in the direction of the target. The assumption is made that the slope of the terrain from the tank to this point is constant. The elevation at this point is sampled and a

vertical angle from the tank to the sampled point is calculated (eq D.7). This is the vertical angle of the tank chassis in the direction of the target with respect to absolute horizontal and provides the offset for the main gun elevation. The desired main gun elevation for the assigned target is arrived at by subtracting the chassis tilt angle from the vertical angle to the target (eq D.8). Figure 7 depicts the various angles used for calculating the main gun elevation. If the desired elevation is below horizontal and

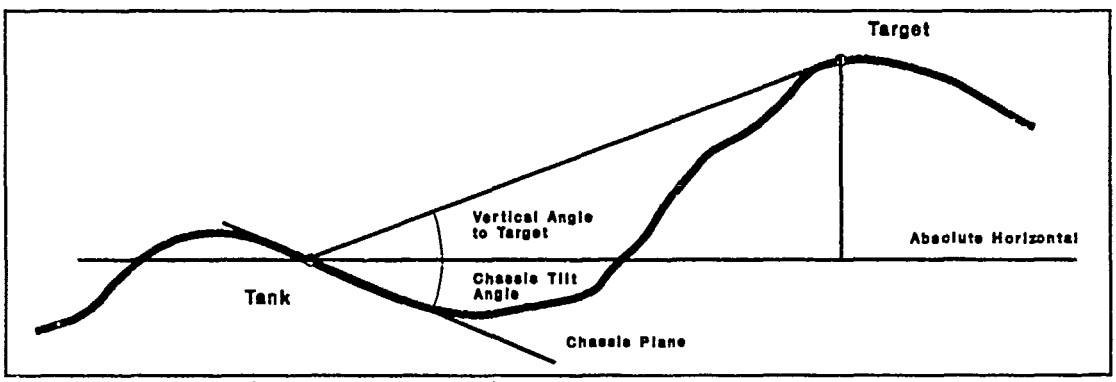

Figure 7: Main Gun Elevation Factors

exceeds the main gun's lower depression limit then the new elevation is set to the lower limit. If the desired elevation is above horizontal and exceeds the main gun's upper elevation limit the new elevation is set to the upper limit. If neither limit is exceeded then the new elevation is set to the desired elevation (eq D.9). The main gun elevation calculation process is formally stated below.

y_{tank} = *tank's y-coord* y_{tgt} = *target's y-coord*
y_{tilt} = *tilt offset y-coord* k_{tilt} = *tilt offset constant*
ϕ_{tgt} = *vertical angle to target* ϕ_{tilt} = *chassis tilt angle*
ϕ_{gun} = *desired gun elevation* ϕ'_{gun} = *new gun elevation*
ϕ_{max} = *gun elevation limit* ϕ_{min} = *gun depression limit*
r = *distance to target*

$$(D.6)\quad \phi_{tgt} = (\arctan \frac{y_{tgt} - y_{tank}}{r} + 360) \bmod 360$$

$$(D.7)\quad \phi_{tilt} = (\arctan \frac{y_{tilt} - y_{tank}}{k_{tilt}} + 360) \bmod 360$$

$$(D.8)\quad \phi_{gun} = (\phi_{tgt} - \phi_{tilt} + 360) \bmod 360$$

$$(D.9)\quad \phi'_{gun} = \begin{cases} \phi_{min} & \text{if } \phi_{min} \le \phi_{gun} \le 180^\circ \\ \phi_{max} & \text{if } 180^\circ \le \phi_{gun} \le \phi_{max} \\ \phi_{gun} & \text{if } (\phi_{min} \le \phi_{gun}) \vee (\phi_{max} \ge \phi_{gun}) \end{cases}$$

3. Line of Sight Calculation

The terrain check is performed by sampling the terrain elevations at specified intervals along the direction to the target. If at any sampled point the elevation exceeds the vertical angle from the firing tank to the target, then the line of sight check will fail. The terrain check process is shown below.

i = *terrain check interval* ϕ_{tgt} = *vert angle to target*
x_s, y_s, z_s = *sample coordinates* θ_{tgt} = *target direction*
x_t, y_t, z_t = *tank's coordinates* r = *range to target*
d = *distance from tank to sample location*

$(D.11)$ *while* $d < r$

$$x_s = (d * \sin \theta_{tgt}) + x_t$$

$z_s = (d * \cos \theta_{tgt}) + z_t$

y_s = *elevation at* (x_s, z_s)

if $y_s > (d * \tan \phi_{tgt}) + y_t$
then line of sight blocked and exit loop
else $d = d + i$ *and continue loop*

APPENDIX E - AF DECISION MAKING MODULE SOURCE CODE

```
;* AF13.CLP

;* 24 FEB 92

;*****************************************************************************
;* This is the NPSNET Autonomous Force decision making program.  It makes all
;* decisions required in a game turn based on the current situation.  This
;* program is called by the main AF program.  The corresponding afinit#.clp
;* program must be called first to initialize the AF.
;*
;* This program can also be run in a stand-alone mode (no interaction with
;* NPSNET or the belief generation program) by running CLIPS then loading in
;* order npsnet#.clp, afinit#.clp, and af#.clp with each file having the same
;* # value.  This capability is for development purposes and allows the
;* program to be run on a PC or any platform that supports CLIPS.
;*
;* This program uses the ordered field notation in its facts.  The commonly
;* used facts are listed below.
;*
;* (tank tank-id# time x-coord y-coord z-coord heading pitch speed gun-dir
;*   gun-elev alive-flag ammo-status fuel-status fire-flag)
;* (plt plt-id# member-id-#s time heading speed x-coord y-coord z-coord)
;* (goal plt-id# G goal1-id goal2-id X goal1-x-coord goal2-x-coord
;*   Z goal1-z-coord goal2-z-coord)
;* (move-order tank-id# time heading from old-x-coord old-y-coord old-z-coord
;*   to new-x-coord new-y-coord new-z-coord at speed)
;* (plt-move plt-id# member-id-#s time plt-heading plt-base-point-heading
;*   speed from x-coord y-coord z-coord to x-coord y-coord z-coord)
;* (target tgt-id time x-coord y-coord z-coord heading speed gun-dir alive fire)
;* (tgt-assign plt-id members-assigned-this-tgt tgt-id tgt-x-coord tgt-y-coord
;*   tgt-z-coord tgt-range tgt-heading tgt-speed)
;* (tgt-data tank-id# tank-pos# tank-x-coord tank-y-coord tank-z-coord
;*   tank-speed turret-pos turret-elev ammo-status tgt-id tgt-range dir-to-tgt
;*   tgt-speed vertical-angle-to-tgt)
;* (globals avg-time ammo-speed game-counter max-coord)
;*
;*****************************************************************************

;* The equipment specific global variables listed below generally reflect M1A1
;* specifications.

(defglobal
   ?*deg_rad* = 0.017453293
   ?*rad_deg* = 57.295777937
   ?*game_counter* = 1          ; counts # of times through loop
   ?*start_time* = 0            ; time at start of battle
   ?*game_time* = 2             ; game time at start of loop
   ?*avg_time* = 1              ; avg time for each decision loop
   ?*cant_const* = 2.0          ; offset for calculating veh cant
   ?*los_interval* = 100        ; interval for line of sight check
   ?*max_speed* = 13            ; meters per second (48 kph, 30 mph)
   ?*turn_rate* = 45.0          ; chassis turn rate in deg per sec
   ?*slew_rate* = 45.0          ; turret slew rate in deg per sec
   ?*wpn_const* = -6e-08        ; weapon accuracy constant
   ?*ammo_speed* = 200.0       ; muzzle velocity meters per sec (set by NPSNET)
```

```
   ?*max_gun_elev* = 60          ; max elev of main gun in degrees
   ?*min_gun_elev* = 300         ; min elev of main gun in degrees
   ?*min_hit_prob* = 50          ; minimum acceptable prob of hit
   ?*max_tgt_range* = 8000       ; max range which target is analyzed
   ?*atk_range* = 5000           ; range at which target is attacked
   ?*stop_atk_range* = 6000      ; range to stop attack
   ?*max_x* = 50000              ; max x coordinate in the world
   ?*max_z* = 50000)             ; max z coordinate in the world

(deffunction dist  "returns distance between two points"
   (?x1 ?z1 ?x2 ?z2)
   (sqrt (+ (** (- ?x2 ?x1) 2)(** (- ?z2 ?z1) 2))))

(deffunction dir  "returns direction in degrees from pt 1 to pt 2"
   (?x1 ?z1 ?x2 ?z2)
   (bind ?dx (- ?x2 ?x1))
   (bind ?dz (- ?z2 ?z1))
   (if (= ?dz 0)                  ;* is dir parallel to x-axis
    then (if (> ?dx 0)
          then (bind ?dir 90.0)
          else (bind ?dir 270.0))
    else (if (> ?dz 0)
          then (bind ?dir (mod (+ (* ?*rad_deg* (atan (/ ?dx ?dz))) 360) 360))
          else (bind ?dir (+ (* ?*rad_deg* (atan (/ ?dx ?dz))) 180)))))

(deffunction vert_dir "returns vert angle in deg from pt 1 to pt 2"
   (?y1 ?range ?y2)
   (bind ?height (- ?y2 ?y1))
   (mod (+ (* ?*rad_deg* (atan (/ ?height ?range))) 360) 360))

(deffunction new_pos   "new pos for a given heading, speed, & time"
   (?x ?z ?heading ?speed ?time)
   (bind ?dist (* ?speed ?time))
   (bind ?nx (+ (* ?dist (sin (* ?*deg_rad* ?heading))) ?x))
   (bind ?nz (+ (* ?dist (cos (* ?*deg_rad* ?heading))) ?z))
   (bind ?ny (get_elev ?nx ?nz))
   (mv-append ?nx ?ny ?nz))

(deffunction form_pos  "returns the coords for the formation pos"
   (?x ?z ?head ?pos#)
   (if (evenp ?pos#)              ;* is tank on left side of plt
      then (bind ?offset-dir 270)  (bind ?offset (* 25 (- ?pos# 1)))
      else (bind ?offset-dir 90)  (bind ?offset (* 25 ?pos#)))
   (bind ?pos-dir (mod (+ ?head ?offset-dir) 360))
   (bind ?px  (+ (* ?offset (sin (* ?*deg_rad* ?pos-dir))) ?x))
   (bind ?pz  (+ (* ?offset (cos (* ?*deg_rad* ?pos-dir))) ?z))
   (bind ?py 0)
   (mv-append ?px ?py ?pz))

(deffunction time_of_flight "returns time of flight to a target"
(?tgt-head ?cur-tgt-dir ?tgt-spd ?miss-spd ?cur-range)
(if (> ?miss-spd ?tgt-spd)
 then (bind ?angle-a (abs (- ?tgt-head ?cur-tgt-dir)))
      (if (> ?angle-a 180)
         then (bind ?angle-a (- 360 ?angle-a)))
  (/ (+ (* ?cur-range ?tgt-spd (cos (deg-rad ?angle-a)))
   (* ?cur-range (sqrt (- (** ?miss-spd 2) (* (** ?tgt-spd 2) (** (sin (deg-rad
                                                          ?angle-a)) 2))))))
   (- (** ?miss-spd 2) (** ?tgt-spd 2)))
 else 1000.0)) ; arbitrarily large value to prevent firing at impossible target
```

```
(deffunction hit_prob    "calculates probability of hitting target"
  (?range ?tgt-spd ?speed)
  (bind ?prob (- (* 100 (exp (* ?*wpn_const* (** ?range 2)))) 5))
  (+ ?prob (* ?tgt-spd -1) (* ?speed -.5)))

(deffunction line_of_sight "checks if l.o.s. exists to a target"
   (?new-dir ?pos# ?tx ?ty ?tz ?range ?tgt-dir ?tgt-elev)
   (bind ?clear yes)
   ;* ensure shot is not blocked by another plt vehicle
   (if (<> ?pos# 3)          ;* is tank the far right vehicle
      then (if (and (> ?new-dir 80) (< ?new-dir 100))
              then (bind ?clear no)))
   (if (<> ?pos# 4)          ;* is tank the far left vehicle
      then (if (and (> ?new-dir 260) (< ?new-dir 280))
              then (bind ?clear no)))
   ;* check if terrain blocks line of sight
   (if (eq ?clear yes)
      then (bind ?dist 50)
           (while (< ?dist ?range)
              (bind ?elev (nth 2 (new_pos ?tx ?tz ?tgt-dir ?dist 1)))
              (if (> ?elev (+ (* ?dist (tan (* ?*deg_rad* ?tgt-elev))) ?ty 2))
                 then (bind ?clear no)  (bind ?dist ?range) )
              (bind ?dist (+ ?dist ?*los_interval*)) ))
   ?clear) ;return the value of ?clear


;*****************************************************************************
;*                            *** Command and Control ***
;*
;* This section performs top level decision making analagous to the command
;* and control functions performed by the commander/staff.  Actions related to
;* communication and coordination between multiple tanks are performed here.
;*****************************************************************************

(defrule start-cycle "reads in the global values and begins cycle"
   (declare (salience 1000))
   ?next <- (initial-fact)
   ?g <- (globals ?time ?ammo-spd ?counter ?max-coord)
=>
   (retract ?g)
   (close)
   (retract ?next)
   (bind ?*avg_time* ?time)
   (bind ?*ammo_speed* ?ammo-spd)
   (bind ?*game_counter* ?counter)
   (bind ?*max_x* ?max-coord)
   (bind ?*max_z* ?max-coord))

(defrule goal-check    "checks if plt has reached its obj"
;* When a plt reaches an immediate goal, the next goal on its list
;* is changed to its immediate goal.  When a plt reaches its final
;* goal it is assigned a new final goal.
   (declare (salience 700))
   (plt ?uid $?mem ?time ?head ?speed ?ux ?uy ?uz)
   ?goal <- (goal ?uid G ?g1 ?g2 X ?gx1&:(<= (abs (- ?gx1 ?ux)) 50) ?gx2
                                 Z ?gz1&:(<= (abs (- ?gz1 ?uz)) 50) ?gz2)
=>
   (if (and (= ?gx1 ?gx2) (= ?gz1 ?gz2))    ;* is it the final goal
    then (printout t "Goal reached." crlf)
         (retract ?goal)
         (bind ?radius (round (/ ?*max_x* 2)))
```

```
        (bind $?pos (new_pos ?radius ?radius (+ ?head 120) (- ?radius 5000) 1))
        (bind ?gx (nth 1 $?pos))  (bind ?gz (nth 3 $?pos))
        (assert (goal ?uid G 0 0 X ?gx ?gx Z ?gz ?gz))
    else (printout t "Detour goal reached." crlf)
        (retract ?goal)
        (assert (goal ?uid G ?g2 ?g2 X ?gx2 ?gx2 Z ?gz2 ?gz2))))

(defrule retract-dead-tgt-goal   "stops attack when tgt destroyed"
   (declare (salience 550))
   (target-destroyed ?tgtid ?tgtx ?tgty ?tgtz)
   ?g <- (goal ?id G ?gid&:(= (* -1 ?tgtid) ?gid) ?g2 X ?gx1 ?gx2 Z ?gz1 ?gz2)
=>
   (retract ?g)
   (assert (goal ?id G ?g2 ?g2 X ?gx2 ?gx2 Z ?gz2 ?gz2)))

(defrule analyze-targets    "creates possible tgt assignments"
   (declare (salience 500))
   (target ?tgtid ?ttime ?tgtx ?tgty ?tgtz ?head ?spd ?gun-dir ?alive ?fire)
   (plt ?uid $?members ?time ?heading ?speed ?ux ?uy ?uz)
=>
   (bind ?range (round (dist ?ux ?uz ?tgtx ?tgtz)))
   (if (< ?range ?*atk_range*)
      then (assert (tgt-assign ?uid $?members ?tgtid ?tgtx ?tgty ?tgtz ?range
                               ?head ?spd)))
   (if (< ?range ?*atk_range*) then (assert (attack-control-fact ?uid)) ))

(defrule reinforcement-call "does plt have 3 or more tgts"
   (declare (salience 485))
   (plt ?uid $?members ?time ?heading ?speed ?ux ?uy ?uz)
   (not (reinforce ?uid ?ux ?uy ?uz))
   (tgt-assign ?uid $?members ?tid1 ?tx1 ?ty1 ?tz1
               ?range1&:(< ?range1 ?*atk_range*) ?head1 ?spd1)
   (tgt-assign ?uid $?members ?tid2&~?tid1 ?tx2 ?ty2 ?tz2
               ?range2&:(< ?range2 ?*atk_range*) ?head2 ?spd2)
   (tgt-assign ?uid $?members ~?tid2&~?tid1 ?tx3 ?ty3 ?tz3
               ?range3&:(< ?range3 ?*atk_range*) ?head3 ?spd3)
=>
   (assert (reinforce ?uid ?ux ?uy ?uz)))

(defrule plt-tgt-assignments  "selects which plt has which tgt"
   (declare (salience 475))
   (tgt-assign ?uid $?mem ?tgtid ?tx ?ty ?tz ?range ?head ?spd)
   ?f <- (tgt-assign ?uid2&~?uid $?mem2 ?tgtid ?tx ?ty ?tz
     ?range2&:(> ?range2 ?range)&:(> ?range2 (/ ?*atk_range* 3.0)) ?head ?spd)
=>
   (retract ?f))

(defrule select-closest-target  "selects 4 tgts closest to plt"
   (declare (salience 450))
   (tgt-assign ?uid $?members ?tid1 ?tx1 ?ty1 ?tz1 ?range1 ?head1 ?spd1)
   (tgt-assign ?uid $?members ?tid2&~?tid1 ?tx2 ?ty2 ?tz2 ?range2 ?head2 ?spd2)
   (tgt-assign ?uid $?members ?tid3&~?tid2&~?tid1 ?tx3 ?ty3 ?tz3
         ?range3 ?head3 ?spd3)
   (tgt-assign ?uid $?members ?tid4&~?tid3&~?tid2&~?tid1 ?tx4 ?ty4
          ?tz4 ?range4 ?head4 ?spd4)
   ?f <- (tgt-assign ?uid $?members ~?tid4&~?tid3&~?tid2&~?tid1
         ?tx5 ?ty5 ?tz5 ?range5 ?head5 ?spd5)
   (test (and (> ?range5 ?range1) (> ?range5 ?range2)
              (> ?range5 ?range3) (> ?range5 ?range4)))
=>
   (retract ?f))
```

```
(defrule reinforce-attack "tasks a plt to reinforce another plt"
   (declare (salience 425))
   (reinforce ?uid ?ux ?uy ?uz)
   (plt ?uid2&~?uid $?members2 ?time2 ?heading2 ?speed2 ?ux2 ?uy2 ?uz2)
   (not (reinforce-control ?uid2))
   (not (tgt-assign ?uid2 $?members2 ?tid ?tx ?ty ?tz
                    ?range&:(< ?range ?*atk_range*) ?head ?spd))
   (not (reinforce ~?uid ?ux3 ?uy3 ?uz3&:(< (dist ?ux2 ?uz2 ?ux3 ?uz3)
                                             (dist ?ux2 ?uz2 ?ux ?uz)) ))
   ?goal <- (goal ?uid2 G ?g1 ?g2 X ?gx1 ?gx2
                  Z ?gz1&:(or (<> ?gx1 ?ux)(<> ?gz1 ?uz)) ?gz2)
=>
   (retract ?goal)
   (assert (goal ?uid2 G ?uid ?g2 X ?ux ?gx2 Z ?uz ?gz2)))

(defrule select-one-reinforcement "select plt for reinforcement"
   (declare (salience 425))
   (reinforce ?uid ?ux ?uy ?uz)
   (goal ?uid1 G ?uid ?g1 X ?ux ?gx1 Z ?uz ?gz1)
   ?g <- (goal ?uid2&~?uid1 G ?uid ?g2 X ?ux ?gx2 Z ?uz ?gz2)
   (plt ?uid1 $?members1 ?time1 ?heading1 ?speed1 ?ux1 ?uy1 ?uz1)
   (plt ?uid2 $?members2 ?time2 ?heading2 ?speed2 ?ux2 ?uy2
      ?uz2&:(< (dist ?ux1 ?uz1 ?ux ?uz)(dist ?ux2 ?uz2 ?ux ?uz)) )
=>
   (retract ?g)
   (assert (reinforce-control ?uid2))
   (assert (goal ?uid2 G ?g2 ?g2 X ?gx2 ?gx2 Z ?gz2 ?gz2)))

(defrule attack-target   "pursues tgt if range < atk_range"
;* If assigned target is within atk_range, the plt orients its
;* movement on the tgt.
   (declare (salience 425))
   ?f <- (attack-control-fact ?uid)  ;* prevents endless loop
   (tgt-assign ?uid $?mem ?tid ?tx ?ty ?tz
        ?range&:(< ?range ?*atk_range*)&:(> ?range 500) ?head ?spd)
   (not (tgt-assign ?uid $?mem2 ~?tid ?tx2 ?ty2 ?tz2
                    ?range2&:(< ?range2 ?range) ?head2 ?spd2))
   ?goal <- (goal ?uid G ?g1 ?g2 X ?gx1 ?gx2
                  Z ?gz1&:(>= (dist ?gx1 ?gz1 ?tx ?tz) 50) ?gz2)
=>
   (retract ?f)
   (retract ?goal)
   (assert (goal ?uid G =(* -1 ?tid) ?g2 X ?tx ?gx2 Z ?tz ?gz2)))

(defrule abandon-reinforcement "stops reinforcement"
   (declare (salience 425))
   ?goal <- (goal ?uid G ?gid ?g2 X ?ux ?gx2 Z ?uz ?gz2)
   (not (reinforce ?gid ?ux1 ?uy1 ?uz1))
   (test (and (> ?gid 0)(< ?gid 12)))
=>
   (retract ?goal)
   (assert (goal ?uid G ?g2 ?g2 X ?gx2 ?gx2 Z ?gz2 ?gz2)))

(defrule abandon-pursuit   "abandons pursuit if tgt too far"
;* If assigned target range is > stop_atk_range, the attack is
;* abandoned.
   (declare (salience 425))
   (tgt-assign ?id $?mem ?tid ?tx ?ty ?tz
               ?range&:(> ?range ?*stop_atk_range*) ?head ?spd)
   ?goal <- (goal ?id G ?gid&:(= (* -1 ?tid) ?gid) ?g2 X ?gx1 ?gx2 Z ?gz1 ?gz2)
=>
```

```
   (retract ?goal)
   (assert (goal ?id G ?g2 ?g2 X ?gx2 ?gx2 Z ?gz2 ?gz2)))

(defrule distribute-fires "assign plt targets to specific tanks"
   (declare (salience 430))
   ?f1 <- (tgt-assign ?uid $?members ?tid1 ?tx1 ?ty1 ?tz1
                      ?range1&:(< ?range1 ?*atk_range*) ?head1 ?spd1)
   ?f2 <- (tgt-assign ?uid $?members2 ?tid2&~?tid1 ?tx2 ?ty2 ?tz2
                      ?range2&:(< ?range2 ?*atk_range*) ?head2 ?spd2)
   (test (subsetp $?members2 $?members))
   (test (> (length $?members2) 1))
   (plt ?uid $?plt-members ?time ?heading ?speed ?ux ?uy ?uz)
=>
   ;********** Determine leftmost & rightmost target **********
   (retract ?f1 ?f2)
   (bind ?t1-dir (dir ?ux ?uz ?tx1 ?tz1)) (bind ?t2-dir (dir ?ux ?uz ?tx2 ?tz2))
   (bind ?t1-angle (- ?heading ?t1-dir)) (bind ?t2-angle (- ?heading ?t2-dir))
   (if (or (and (<= (abs ?t1-angle) 180) (<= (abs ?t2-angle) 180))
           (and (> (abs ?t1-angle) 180) (> (abs ?t2-angle) 180)) )
      ;* both tgt angles straddle 0 degree line or both do not
      then (if (<= ?t1-angle ?t2-angle)
               then (bind ?t1 right) (bind ?t2 left)
               else (bind ?t1 left) (bind ?t2 right) )
      ;* one tgt angle straddles 0 degree line, the other does not
      else (if (<= ?t1-angle ?t2-angle)
               then (bind ?t1 left) (bind ?t2 right)
               else (bind ?t1 right) (bind ?t2 left) ))

   ;******* Assign left target to leftmost tank(s), right to rightmost *******
   (if (= (length $?members2) 4)    ;* is it a full plt's worth of tanks *
      then (if (eq ?t1 left)
            then (assert (tgt-assign ?uid =(nth 2 $?members) =(nth 4 $?members)
                                     ?tid1 ?tx1 ?ty1 ?tz1 ?range1 ?head1 ?spd1))
                 (assert (tgt-assign ?uid =(nth 1 $?members) =(nth 3 $?members)
                                     ?tid2 ?tx2 ?ty2 ?tz2 ?range2 ?head2 ?spd2))
            else (assert (tgt-assign ?uid =(nth 1 $?members) =(nth 3 $?members)
                                     ?tid1 ?tx1 ?ty1 ?tz1 ?range1 ?head1 ?spd1))
                 (assert (tgt-assign ?uid =(nth 2 $?members) =(nth 4 $?members)
                                     ?tid2 ?tx2 ?ty2 ?tz2 ?range2 ?head2 ?spd2)) )
      else ;* there are only two tanks being considered *
           (bind ?mem1 (nth 1 $?members2)) (bind ?mem2 (nth 2 $?members2))
           (if (eq ?t1 left)
              then (if (oddp ?mem1) ;* are tanks on the right side of plt *
                       then (assert (tgt-assign ?uid ?mem1 ?tid1 ?tx1 ?ty1 ?tz1
                                                ?range1 ?head1 ?spd1))
                            (assert (tgt-assign ?uid ?mem2 ?tid2 ?tx2 ?ty2 ?tz2
                                                ?range2 ?head2 ?spd2))
                       else (assert (tgt-assign ?uid ?mem2 ?tid1 ?tx1 ?ty1 ?tz1
                                                ?range1 ?head1 ?spd1))
                            (assert (tgt-assign ?uid ?mem1 ?tid2 ?tx2 ?ty2 ?tz2
                                                ?range2 ?head2 ?spd2)) )
                 else (if (oddp ?mem1)
                       then (assert (tgt-assign ?uid ?mem2 ?tid1 ?tx1 ?ty1 ?tz1
                                                ?range1 ?head1 ?spd1))
                            (assert (tgt-assign ?uid ?mem1 ?tid2 ?tx2 ?ty2 ?tz2
                                                ?range2 ?head2 ?spd2))
                       else (assert (tgt-assign ?uid ?mem1 ?tid1 ?tx1 ?ty1 ?tz1
                                                ?range1 ?head1 ?spd1))
                            (assert (tgt-assign ?uid ?mem2 ?tid2 ?tx2 ?ty2 ?tz2
                                                ?range2 ?head2 ?spd2)) ))))
```

```
;*****************************************************************************
;*                              *** Movement ***
;*
;* This section develops platoon and individual tank moves based assigned
;* march objective.
;*****************************************************************************

(defrule plt-move   "calculates platoon move"
;* This rule calculates the direction to the platoon march objective and any
;* platoon turns required to move toward the obj.  It also determines the
;* platoon's new speed.
   (declare (salience 415))
   (plt ?uid $?members ?time ?heading ?speed ?ux ?uy ?uz)
   (goal ?uid G ?g1 ?g2 X ?gx1 ?gx2 Z ?gz1 ?gz2)
=>
   (bind ?azimuth (dir ?ux ?uz ?gx1 ?gz1))

   ;*** Calculate unconstrained turn ***
   (bind ?turn-angle (- ?azimuth ?heading))
   (if (> (abs ?turn-angle) 180)
   ;* do azimuth and heading straddle the 0 degree line *
      then (if (> ?turn-angle 0)
              then (bind ?turn-angle (- ?turn-angle 360))
              else (bind ?turn-angle (+ ?turn-angle 360)) ))

   ;*** Calculate platoon speed ***
   (if (> (abs ?turn-angle) 1)  ;is platoon turning
      then (if (>= ?speed 5.0)
              then (bind ?speed 7.0)
              else (bind ?speed (* ?speed 1.25)) )
      else (if (<= ?speed 8.0)
              then (bind ?speed (* ?speed 1.25))
              else (bind ?speed 10.0) ))
   (if (and (<> ?g1 0) (< (dist ?ux ?uz ?gx1 ?gz1) 500) (> ?speed 1.0))
   ;* is platoon close to attack target *
      then (bind ?speed (* ?speed .75)) )

   ;*** Constrain turn ***
   (bind ?udist (* ?speed ?*avg_time*))
   (bind ?max-dist (* ?udist (/ 175 100)) ;max dist outside tank can travel
   (bind ?max-turn (* 2 ?*rad_deg* (asin (/ ?max-dist (* 2 175)))))
   (if (> ?max-turn ?*turn_rate*) then (bind ?max-turn ?*turn_rate*))
   (if (> (abs ?turn-angle) ?max-turn)
      then (if (> ?turn-angle 0)
              then (bind ?turn-angle ?max-turn)
              else (bind ?turn-angle (* -1 ?max-turn)) ))
   (bind ?plt-turn (/ ?turn-angle 2))
   (bind ?base-dir (mod (+ ?heading ?plt-turn 360) 360))
   (bind ?azimuth (mod (+ ?heading ?turn-angle 360) 360))
   (bind ?nx (+ (* ?udist (sin (* ?*deg_rad* ?base-dir))) ?ux))
   (bind ?nz (+ (* ?udist (cos (* ?*deg_rad* ?base-dir))) ?uz))
   (assert (plt-move ?uid $?members ?time ?azimuth ?base-dir ?speed from
              ?ux ?uy ?uz to ?nx =(get_elev ?nx ?nz) ?nz)))

(defrule tank-move  "calculates tank moves required to stay in formation"
;* Given a platoon move order, this rule calculates the steering and speed
;* commands necessary to stay in formation.
   (declare (salience 410))
   (plt-move ?uid $?members ?time ?plt-az ?base-dir ?spd from
             ?ux ?uy ?uz to ?nx ?ny ?nz)
```

```
   (tank ?id&:(member ?id $?members) ?time2 ?tx ?ty ?tz ?heading ?pitch
             ?speed ?gun-dir ?gun-elev ?alive ?ammo ?fuel ?fire)
=>
   ;*** Calculate coordinates of next formation position ***
   (bind ?pos# (member ?id $?members))
   (bind $?form-pos (form_pos ?nx ?nz ?plt-az ?pos#))
   (bind ?fx (nth 1 $?form-pos))
   (bind ?fz (nth 3 $?form-pos))

   ;*** Calculate new tank heading ***
   (bind ?azimuth (dir ?tx ?tz ?fx ?fz))
   (bind ?turn (- ?azimuth ?heading))
   (if (> (abs ?turn) 180) ;* does turn bracket 0 degree line
      then (if (> ?turn 0)
             then (bind ?turn (- ?turn 360))
        else (bind ?turn (+ ?turn 360)) ))
   (if (> (abs ?turn) ?*turn_rate*) ;* is turn too sharp
      then (if (> ?turn 0)
             then (bind ?turn ?*turn_rate*)
             else (bind ?turn (* -1 ?*turn_rate*)) ))
   (bind ?azimuth (mod (+ ?heading ?turn 360) 360))

   ;*** Calculate tank speed ***
   (bind ?new-spd (/ (dist ?tx ?tz ?fx ?fz) ?*avg_time*))
   (bind ?spd-chg (- ?new-spd ?speed))
   (if (> ?speed 0)
       then (if (> (/ (abs ?spd-chg) ?speed) 0.5)
              then (if (< ?spd-chg 0)
                     then (bind ?new-spd (* ?speed 0.5))
                     else (bind ?new-spd (* ?speed 1.5)) ))
       else (bind ?new-spd 0.0) )
   (if (> ?new-spd 20.0) then (bind ?new-spd 20.0)) ;* speed governor
   (if (<= ?fuel 0.0) then (bind ?new-spd 0.0))

   (assert (move-order ?id ?time2 ?azimuth from ?tx ?ty ?tz to
           =(new_pos ?tx ?tz ?azimuth ?new-spd ?*avg_time*) at ?new-spd)))


;*****************************************************************************
;*                        *** Evaluate Move Orders ***
;*
;* This section verifies each move order to ensure that it is O.K.
;*****************************************************************************

(defrule collision-avoidance   "ensures friendly tanks don't collide"
;* If 2 tanks are going to collide, this rule will cause the tank on the left
;* to steer left and the tank on the right to steer right.
   (declare (salience 1500))
   ?m1 <- (move-order ?id ?time ?azimuth from ?tx ?ty ?tz to ?nx ?ny ?nz at ?s)
   ?m2 <- (move-order ?id2&~?id ?time2 ?azimuth2 from ?tx2 ?ty2 ?tz2 to
                      ?nx2 ?ny2 ?nz2 at ?s2)
   (not (no collision-avoidance loop ?id ?id2))
   (not (no collision-avoidance loop ?id2 ?id))
   (test (<= (dist ?nx ?nz ?nx2 ?nz2) 25.0))
=>
   (retract ?m1 ?m2)
   (bind ?dir (dir ?nx ?nz ?nx2 ?nz2)) ; dir from tank1 to tank2
   (bind ?angle-d (- ?dir ?azimuth))
   (if (> (abs ?angle-d) 180)
   ;* does angle straddle the 0 degree line *
      then (if (> ?angle-d 0)
```

```
            then (bind ?angle-d (- ?angle-d 360))
            else (bind ?angle-d (+ ?angle-d 360)) ))
   (if (< ?angle-d 0) ; is tank2 left of tank1
      then (assert (move-order ?id ?time =(mod (+ ?azimuth 20) 360) from
                               ?tx ?ty ?tz to ?nx ?ny ?nz at ?s))
           (assert (move-order ?id2 ?time2 =(mod (+ ?azimuth2 -20 360) 360) from
                               ?tx2 ?ty2 ?tz2 to ?nx2 ?ny2 ?nz2 at ?s2))
      else (assert (move-order ?id ?time =(mod (+ ?azimuth -20 360) 360) from
                               ?tx ?ty ?tz to ?nx ?ny ?nz at ?s))
           (assert (move-order ?id2 ?time2 =(mod (+ ?azimuth2 20) 360) from
                               ?tx2 ?ty2 ?tz2 to ?nx2 ?ny2 ?nz2 at ?s2)) )
   (assert (no collision-avoidance loop ?id ?id2)) )


;*****************************************************************************
;*                        *** Turret Actions ***
;*
;* This section makes the decisions related to the aiming and firing of the
;* tank's main gun.  The aiming(scanning) decisions are analogous to the
;* turret slewing decision made by the gunner to orient the main gun.  The
;* firing decision is analagous to the tank commander's firing decision.
;*****************************************************************************

(defrule aim-at-assigned-target "generates turret order for assigned tgt"
;* Determines the best turret movement for an assigned tgt
   (declare (salience -475))
   (tgt-assign ?uid $?assign-mems ?tgtid ?tgtx ?tgty ?tgtz ?range ?tgt-head
               ?tgt-spd)
   (plt ?uid $?members ?utime ?uhead ?uspd ?ux ?uy ?uz)
   (move-order ?id&:(member ?id $?assign-mems) ?time ?azimuth from ?tx ?ty ?tz
            to ?nx ?ny ?nz at ?spd)
   (tank ?id ?time ?tx ?ty ?tz ?heading ?pitch ?speed ?gun-dir ?gun-elev
               ?alive ?ammo ?fuel ?fire)
=>
   ;********** calculate target position at time of impact **********
   (bind ?cur-range (dist ?tx ?tz ?tgtx ?tgtz))
   (bind ?missile-speed (+ ?*ammo_speed* ?spd))
   (bind ?cur-tgt-dir (dir ?tx ?tz ?tgtx ?tgtz))    ;* current dir to target *
   (bind ?time-of-flt (time_of_flight ?tgt-head ?cur-tgt-dir ?tgt-spd
                                      ?missile-speed ?cur-range))
   (bind ?lead-time (+ ?time-of-flt (/ ?*avg_time* 1.0)))
   (bind $?tgt-pos (new_pos ?tgtx ?tgtz ?tgt-head ?tgt-spd ?lead-time))
   (bind ?tgtx (nth 1 $?tgt-pos))
   (bind ?tgty (nth 2 $?tgt-pos))
   (bind ?tgtz (nth 3 $?tgt-pos))

   ;********** calculate turret deflection **********
   (bind ?tgt-dir (dir ?tx ?tz ?tgtx ?tgtz))             ;* dir to target *
   (bind ?turret-dir (mod (+ ?azimuth ?gun-dir) 360))    ;* turret's dir *
   (bind ?tgt-angle (- ?tgt-dir ?turret-dir))            ;* gun-tgt angle *
   (bind ?max-slew (* ?*avg_time* ?*slew_rate*))
   (if (<= (mod (+ ?max-slew (abs ?tgt-angle)) 360) (* ?max-slew 2))
   ;* can turret slew all the way to target during one decision cycle *
    then (bind ?new-dir (mod (+ ?tgt-angle ?gun-dir 360) 360))
    else (if (>= (mod (+ ?tgt-angle 360) 360) 180)
          then ;* slew turret left toward target *
               (bind ?new-dir (mod (- (+ ?gun-dir 360) ?max-slew) 360))
          else ;* slew turret right toward target *
               (bind ?new-dir (mod (+ ?gun-dir ?max-slew) 360)))))
```

```
   ;********** calculate main gun elevation **********
   ;* calculate tank chassis tilt toward target *
   (bind $?rise_pos (new_pos ?tx ?tz ?tgt-dir ?*cant_const* 1))
   (bind ?rise (get_elev (nth 1 $?rise_pos) (nth 3 $?rise_pos)))
   (bind ?cant (vert_dir ?ty ?*cant_const* ?rise))

   ;* calculate vertical angle to target *
   (bind ?range (dist ?tx ?tz ?tgtx ?tgtz))
   (bind ?tgt-elev (vert_dir ?ty ?range ?tgty))

   ;* adjust next gun elev to account for vehicle tilt *
   (bind ?new-elev (mod (+ (- ?tgt-elev ?cant) 360) 360))
   (bind ?rel-elev ?new-elev)    ;* without consideration of physical limits *
   (bind ?rel-dir (mod (+ ?tgt-angle ?gun-dir 360) 360))
   (if (and (> ?new-elev 180)(< ?new-elev ?*min_gun_elev*))
   ;* is desired main gun elev beyond physical limits *
       then (bind ?new-elev ?*min_gun_elev*)          ;* below max depression *
       else (if (and (> ?new-elev ?*max_gun_elev*)(< ?new-elev 180))
                then (bind ?new-elev ?*max_gun_elev*)))    ;* above max elev *
   (assert (tgt-data ?id =(member ?id $?members) ?tx ?ty ?tz ?spd ?rel-dir
                  ?rel-elev ?ammo ?tgtid ?range ?tgt-dir ?tgt-spd ?tgt-elev))
   (assert (turret-order ?id ?new-dir ?new-elev 0 ?tgtid)))

(defrule scan-for-targets   "search for tgts if none is assigned"
;* Returns turret to assigned sector for observation.
   (declare (salience -475))
   (plt-move ?uid $?members ?utime ?azimuth ?base-dir ?uspd from
             ?ux ?uy ?uz to ?nx ?ny ?nz)
   (tank ?id&:(member ?id $?members) ?time ?tx ?ty ?tz ?heading
             ?pitch ?speed ?gun-dir ?gun-elev ?alive ?ammo ?fuel ?old-fire)
   (not (turret-order ?id ?new_dir ?new_elev ?fire ?tgtid))
=>
   (bind ?pos# (member ?id $?members))
   (if (evenp ?pos#)
   ;* is tank on right side of plt
      then (bind ?offset (* -22.5 (- (- ?pos# 1) 1)))
      else (bind ?offset (* 22.5 (- ?pos# 1))))
   (bind ?sector-pos (mod (+ ?offset 360) 360))
   (bind ?slew-angle (- ?sector-pos ?gun-dir))
   (bind ?max-slew  (* ?*avg_time* ?*slew_rate*))
   (if (<= (mod (+ ?max-slew (abs ?slew-angle)) 360) (* ?max-slew 2))
   ;* can turret slew to assigned sector during next decision cycle
      then (bind ?new-dir ?sector-pos)
      else (if (>= (mod (+ ?slew-angle 360) 360) 180)
             then ;* slew turret left toward sector
                  (bind ?new-dir (mod (- (+ ?gun-dir 360) ?max-slew) 360))
             else ;* slew turret right toward sector
                  (bind ?new-dir (mod (+ ?gun-dir ?max-slew) 360))))
   (assert (turret-order ?id ?new-dir 0 0 0)))

(defrule check-fire "disable firing if friendly tank is in the way"
   (declare (salience -495))
   ?data <- (tgt-data ?id ?pos# ?tx ?ty ?tz ?spd ?new-dir ?new-elev ?ammo
                      ?tgtid ?range ?tgt-dir ?tgt-spd ?tgt-elev)
   (turret-order ?id ?new-dir ?new-elev 0 ?tgtid)
   (tank ?id2&~?id ?time ?x2 ?y2 ?z2 $?restoffact)
   (test (<= (abs (- (dir ?tx ?tz ?x2 ?z2) ?tgt-dir))
             (* ?*rad_deg* (atan (/ 45.0 ?range)))))
   (test (< (dist ?tx ?tz ?x2 ?z2) ?range))
=>
   (retract ?data))
```

```
(defrule engage-target        "makes firing decision"
;* Determines if target is engageable and if probability of hit is acceptable.
;* If "fire" decision is made, turret-order is retracted and reasserted with
;* fire flag on.
   (declare (salience -500))
   ?data <- (tgt-data ?id ?pos# ?tx ?ty ?tz ?spd ?new-dir ?new-elev ?ammo
                      ?tgtid ?range ?tgt-dir ?tgt-spd ?tgt-elev)
   ?tur-order <- (turret-order ?id ?new-dir ?new-elev 0 ?cgtid)
;   (test (> ?ammo 0))
   (test (>= (hit_prob ?range ?tgt-spd ?spd) ?*min_hit_prob*))
=>
   (retract ?data)
   bind ?los (line_of_sight ?new-dir ?pos# ?tx ?ty ?tz ?range ?tgt-dir
                                                              ?tgt-elev))
   (if (eq ?los yes)
   ;* does tank have line of sight with target
      then (retract ?tur-order)
           (bind ?fire 1)
           (assert (turret-order ?id ?new-dir ?new-elev 1 ?tgtid)) ))

(defrule send-orders  "transmits orders and saves new info in C data structs"
   (declare (salience -9000))
   (move-order ?id ?time ?azimuth from ?tx ?ty ?tz to ?nx ?ny ?nz at ?speed)
   (turret-order ?id ?new-dir ?new-elev ?fire ?tgtid)
   (tank ?id ?time ?tx ?ty ?tz ?heading ?pitch ?old-spd ?gun-dir ?gun-elev
            ?alive ?ammo ?fuel ?old-fire)
   (test (or (> (abs (- ?azimuth ?heading)) .1)
             (> (abs (- ?speed ?old-spd)) .1) (= ?fire 1)
             (<> ?new-dir ?gun-dir) (<> ?new-elev ?gun-elev)))
=>
   (bind ?vehno (+ 190 (- ?id 1)))
   (bind ?ty (get_elev ?tx ?tz))
   (send_order ?time ?id ?tx ?ty ?tz ?azimuth ?new-dir ?new-elev ?speed ?fire)
   (save_saf ?id ?azimuth ?speed ?new-dir ?new-elev ?fire));

(defrule save-plt-info  "stores plt info in C data structures for next turn"
   (declare (salience -9800))
   (plt ?uid $?members ?time ?head ?spd ?ux ?uy ?uz)
   (plt-move ?uid $?members ?time ?plt-az ?base-dir ?new-spd from
            ?ux ?uy ?uz to ?nx ?ny ?nz)
   (test (or (<> ?head ?plt-az) (<> ?spd ?new-spd)))  ;* save only if changed
=>
   (save_plts ?uid ?base-dir ?new-spd))

(defrule save-goals  "stores goal info in C data structures for next turn"
   (declare (salience -9810))
   (goal ?id G ?gid1 ?gid2 X ?gx1 ?gx2 Z ?gz1 ?gz2)
=>
   (save_goals ?id ?gid1 ?gid2 ?gx1 ?gx2 ?gz1 ?gz2))
```

LIST OF REFERENCES

1. Robinson, C. A., Jr., "Simulation Improves Desert Combat Operations," *Signal*, Armed Forces Communications and Electronics Association, pp. 23-27, July 1991.

2. Zyda, M. J., and Pratt, D. R., "NPSNET: A 3D Visual Simulator for Virtual World Exploration and Experimentation," *Digest of Technical Papers XXII*, SID International Symposium, 8 May 1991.

3. Kaelbling, L. P., and Rosenschein S. J., "Action and Planning in Embedded Agents," in *Designing Autonomous Agents Theory and Practice from Biology to Engineering and Back*, MIT Press, 1991.

4. Bhargava, H. K., and Branley, W. C., "On Transforming Knowledge to Belief in a Combat Simulation System," unpublished NPS working paper, 30 August 1991.

5. Branley, W. C., *Modeling Observation in Intelligent Agents: Knowledge and Belief*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1992.

6. Maes, P., *Designing Autonomous Agents Theory and Practice from Biology to Engineering and Back*, MIT Press, 1991.

7. Agre, P. E., and Chapman, D., "What Are Plans for?," in *Designing Autonomous Agents Theory and Practice from Biology to Engineering and Back*, MIT Press, 1991.

8. United States Army Armor Center, *Characteristics and Description Book M1A1 Tank*, Fort Knox, Kentucky.

9. Combined Arms and Services Staff School, *Military Decision Making*, Fort Leavenworth, Kansas, January 1986.

10. United States Army Command and General Staff College, *Field Manual 101-5, Staff Organization and Operations*, Fort Leavenworth, Kansas, 25 May 1984.

11. Zyda, M. J., and others, "NPSNET: Constructing a 3D Virtual World," Proceedings of the 1992 Symposium on Interactive 3D Graphics, 29 March - 1 April 1992.

12. University of Central Florida, Institute for Simulation and Training, *Game for Intelligent Simulated Military Opponents Game Description.*

13. Lyndon B. Johnson Space Center, Software Technology Branch, *CLIPS Reference Manual, Volume II, Advanced Programming Guide*, sec. 3-5, 11 January 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. Library, Code 52 — 2
   Naval Postgraduate School
   Monterey, California 93943-5002

3. Department Chairman, Code AS — 1
   Department of Administrative Sciences
   Naval Postgraduate School
   Monterey, California 93943-5000

4. Professor Hemant K. Bhargava, Code AS/Bh — 2
   Administrative Sciences Department
   Naval Postgraduate School
   Monterey, California 93943-5000

5. Mr. David R. Pratt, Code CS/Pr — 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5000

6. Professor Michael J. Zyda, Code CS/Zk — 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5000

7. Professor Se-Hung Kwak, Code CS/Kw — 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5000

8. Artificial Intelligence Center — 1
   HQDA, OCSA
   ATTN: DACS-DMA
   Pentagon, RM 1D659
   Washington DC 20310-0200

9. Defense Advanced Research Projects Agency — 1
   ATTN: MAJ David Neyland
   3701 N. Fairfax Drive
   Arlington, VA 22203-1714

10. CPT Michael E. Culpepper 1
Rt. 1 Box 32
Malvern, AR 72104

11. CPT William C. Branley 1
P.O. Box 3658
Merrifield, VA 22116-3658